**1. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

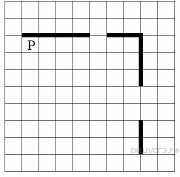
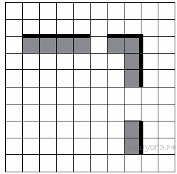
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с верх­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но под го­ри­зон­таль­ной сте­ной у её ле­во­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но ниже го­ри­зон­таль­ной стены и левее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет мак­си­маль­ное число, крат­ное 5. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 5. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — максимальное число, крат­ное 5.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 10 25 12 | 25 |

**2. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

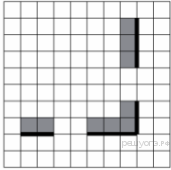
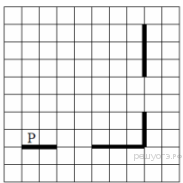
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но над го­ри­зон­таль­ной сте­ной у её ле­во­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше го­ри­зон­таль­ной стены и левее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму чисел, крат­ных 6. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 6. Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число — сумму чисел, крат­ных 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 12 25 6 | 18 |

**3. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

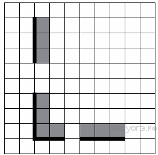
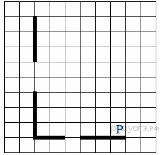
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Левый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но над го­ри­зон­таль­ной сте­ной у её левого конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше го­ри­зон­таль­ной стены и правее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, крат­ных 4. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 4. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — количество чисел, крат­ных 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 16 26 24 | 2 |

**4. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

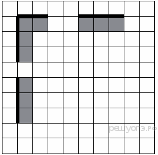
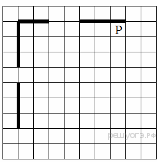
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Левый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но под го­ри­зон­таль­ной сте­ной у её пра­во­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но нижего­ри­зон­таль­ной стены и левее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ми­ни­маль­ное число, крат­ное 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — минимальное число, крат­ное 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 21 12 31 | 12 |

**5.**Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

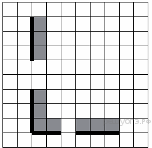
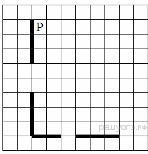
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но спра­ва от вер­ти­каль­ной стены у её верх­не­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше го­ри­зон­таль­ной стены и правее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму чисел, крат­ных 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число — сумму чисел, крат­ных 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 12 25 9 | 21 |

**6. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

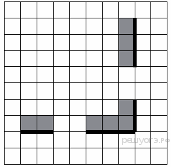
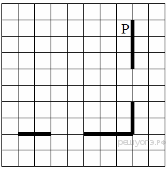
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но слева от вер­ти­каль­ной стены у её верх­не­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше го­ри­зон­таль­ной стены и левее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, крат­ных 6. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 18 26 24 | 2 |

**7. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

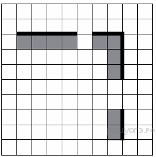
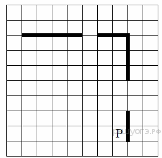
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В каж­дой стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но слева от вер­ти­каль­ной стены у её ниж­не­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше го­ри­зон­таль­ной стены и левее вер­ти­каль­ной стены. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет мак­си­маль­ное число, крат­ное 4. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 4. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — максимальное число, крат­ное 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 8 16 11 | 16 |

**8. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

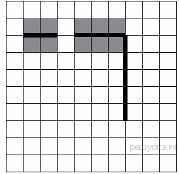
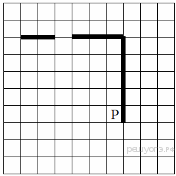
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В го­ри­зон­таль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной рядом с вер­ти­каль­ной сте­ной слева от её ниж­не­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше и ниже го­ри­зон­таль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму чисел, окан­чи­ва­ю­щих­ся на 4. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 4. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — сумму чисел, окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3  14  25  24 | 38 |

**9. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

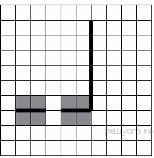
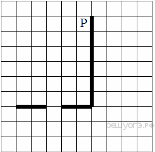
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В го­ри­зон­таль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны.Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной рядом с вер­ти­каль­ной сте­ной слева от её верх­не­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше и ниже го­ри­зон­таль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, окан­чи­ва­ю­щих­ся на 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — количество чисел, окан­чи­ва­ю­щих­ся на 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 13 23 24 | 2 |

**10. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

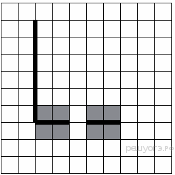
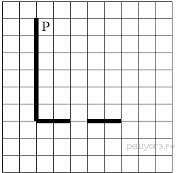
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В го­ри­зон­таль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной рядом с вер­ти­каль­ной сте­ной спра­ва от её верх­не­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но выше и ниже го­ри­зон­таль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле. На­зва­ние файла и ка­та­лог для со­хра­не­ния Вам со­об­щат ор­га­ни­за­то­ры экзамена.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет мак­си­маль­ное число, окан­чи­ва­ю­ще­е­ся на 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — максимальное число, окан­чи­ва­ю­ще­е­ся на 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 13 23 3 | 23 |

**11. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

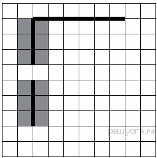
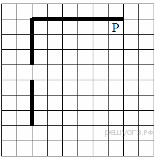
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Левый конец го­ри­зон­таль­ной стены соединён с верх­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В го­ри­зон­таль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но под го­ри­зон­таль­ной сте­ной у её пра­во­го конца. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но левее и пра­вее вер­ти­каль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки,

удовлетворяющие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ми­ни­маль­ное число, окан­чи­ва­ю­ще­е­ся на 6. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 6. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — минимальное число, окан­чи­ва­ю­ще­е­ся на 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 26 16 36 | 16 |

**12. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

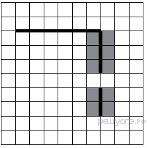
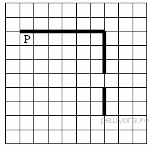
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с верх­ним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В вер­ти­каль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны.Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но под го­ри­зон­таль­ной сте­ной у её ле­во­го конца.На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но левее и пра­вее вер­ти­каль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму чисел, окан­чи­ва­ю­щих­ся на 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число — сумму чисел, окан­чи­ва­ю­щих­ся на 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 13 23 24 | 36 |

**13. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

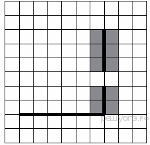
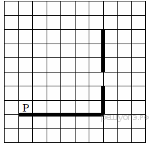
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с нижним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В вер­ти­каль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны.Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но над го­ри­зон­таль­ной сте­ной у её ле­во­го конца.На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но левее и пра­вее вер­ти­каль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, окан­чи­ва­ю­щих­ся на 6. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 6. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — ко­ли­че­ство чисел, окан­чи­ва­ю­щих­ся на 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 16 26 24 | 2 |

**14. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

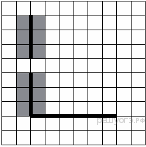
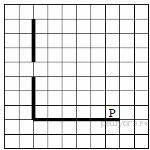
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Левый конец го­ри­зон­таль­ной стены соединён с нижним кон­цом вер­ти­каль­ной стены. Длины стен неизвестны. В вер­ти­каль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны.Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но над го­ри­зон­таль­ной сте­ной у её пра­во­го конца.На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но левее и пра­вее вер­ти­каль­ной стены. Про­ход дол­жен остать­ся незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ми­ни­маль­ное число, окан­чи­ва­ю­ще­е­ся на 4. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 4. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — ми­ни­маль­ное число, окан­чи­ва­ю­ще­е­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 24 14 34 | 14 |

**15. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

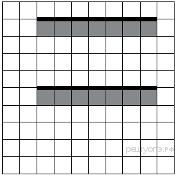
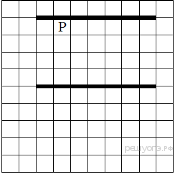
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся две оди­на­ко­вые го­ри­зон­таль­ные па­рал­лель­ные стены, рас­по­ло­жен­ные друг под дру­гом и от­сто­я­щие друг от друга более чем на 1 клетку. Левые края стен на­хо­дят­ся на одном уровне. Длины стен неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но под верх­ней стеной.На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные ниже го­ри­зон­таль­ных стен. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму чисел, крат­ных 5. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 5. Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число — сумму чисел, кратных 5.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 15 25 6 | 40 |

**16. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

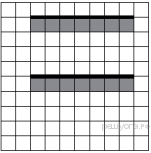
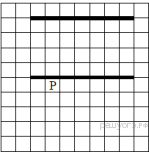
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся две оди­на­ко­вые го­ри­зон­таль­ные па­рал­лель­ные стены, рас­по­ло­жен­ные друг под дру­гом и от­сто­я­щие друг от друга более чем на 1 клетку. Левые края стен на­хо­дят­ся на одном уровне. Длины стен неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но под ниж­ней стеной.На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные ниже го­ри­зон­таль­ных стен. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, крат­ных 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, крат­ное 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число — ко­ли­че­ство чисел, крат­ных 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3  12  26  24 | 2 |

**17. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

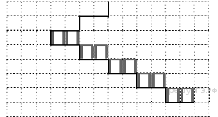
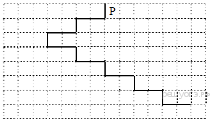
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся лестница. Сна­ча­ла лест­ни­ца спус­ка­ет­ся вниз спра­ва налево, затем спус­ка­ет­ся вниз слева направо. Вы­со­та каж­дой ступени — одна клетка, ширина — две клетки. Робот на­хо­дит­ся спра­ва от верх­ней сту­пе­ни лестницы. Ко­ли­че­ство ступенек, ве­ду­щих влево, и ко­ли­че­ство ступенек, ве­ду­щих вправо, неизвестно. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния лест­ни­цы и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но над сту­пе­ня­ми лестницы, спус­ка­ю­щей­ся слева направо. Тре­бу­ет­ся за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

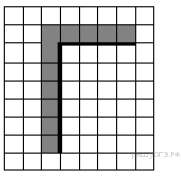
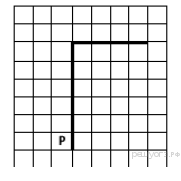
**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет опре­де­ля­ет сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — признак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 14 24 144 22 12 0 | 168 |

**18. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1**



На бес­ко­неч­ном поле име­ет­ся вер­ти­каль­ная стена. Длина стены неизвестна. От верх­не­го конца стены впра­во от­хо­дит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной слева от ниж­не­го края вер­ти­каль­ной стены.

На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные левее вер­ти­каль­ной стены и выше го­ри­зон­таль­ной стены и при­ле­га­ю­щие к ним. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм на­пи­ши­те в тек­сто­вом ре­дак­то­ре и со­хра­ни­те в тек­сто­вом файле. На­зва­ние файла и ка­та­лог для со­хра­не­ния Вам со­об­щат ор­га­ни­за­то­ры экзамена.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти целых чисел опре­де­ля­ет сумму двух наи­боль­ших и сумму двух наименьших. Про­грам­ма долж­на вы­ве­сти две этих суммы в ука­зан­ном порядке. Про­грам­ма по­лу­ча­ет на вход целые числа, ко­ли­че­ство введённых чисел не известно, после­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000.

**19. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

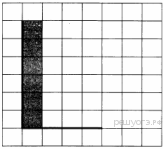
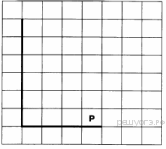
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся вер­ти­каль­ная стена. Длина стены неизвестна. От ниж­не­го конца стены впра­во от­хо­дит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной над пра­вым краем го­ри­зон­таль­ной стены. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные пра­вее вер­ти­каль­ной стены и при­мы­ка­ю­щие к ней. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите про­грам­му для ре­ше­ния сле­ду­ю­щей задачи. Ка­ме­ра на­блю­де­ния ре­ги­стри­ру­ет в ав­то­ма­ти­че­ском ре­жи­ме ско­рость про­ез­жа­ю­щих мимо неё автомобилей, округ­ляя зна­че­ния ско­ро­сти до целых чисел. Не­об­хо­ди­мо опре­де­лить мак­си­маль­ную за­ре­ги­стри­ро­ван­ную ско­рость автомобиля. Если ско­рость хотя бы од­но­го ав­то­мо­би­ля была мень­ше 30 км/ч, вы­ве­ди­те «YES», иначе вы­ве­ди­те «N0».

Программа по­лу­ча­ет на вход число про­ехав­ших ав­то­мо­би­лей N (1 < N < 30), затем ука­зы­ва­ют­ся их скорости. Зна­че­ние ско­ро­сти не может быть мень­ше 1 и боль­ше 300.Программа долж­на сна­ча­ла вы­ве­сти мак­си­маль­ную скорость, затем YES или NO.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 74 69 63 66 | 74 NO |

**20.**Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

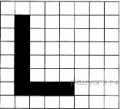
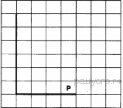
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся вер­ти­каль­ная стена. Длина стены неизвестна. От ниж­не­го конца стены впра­во от­хо­дит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной над пра­вым краем го­ри­зон­таль­ной стены. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные пра­вее вер­ти­каль­ной стены, выше го­ри­зон­таль­ной стены и при­мы­ка­ю­щие к ним. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите про­грам­му для ре­ше­ния сле­ду­ю­щей задачи. Ка­ме­ра на­блю­де­ния ре­ги­стри­ру­ет в ав­то­ма­ти­че­ском ре­жи­ме ско­рость про­ез­жа­ю­щих мимо неё автомобилей, округ­ляя зна­че­ния ско­ро­сти до целых чисел. Не­об­хо­ди­мо опре­де­лить ми­ни­маль­ную за­ре­ги­стри­ро­ван­ную ско­рость автомобиля. Если ско­рость хотя бы од­но­го ав­то­мо­би­ля была боль­ше 80 км/ч, вы­ве­ди­те «YES», иначе вы­ве­ди­те «NO».

Программа по­лу­ча­ет на вход число про­ехав­ших ав­то­мо­би­лей N (1 =< N =< 30), затем ука­зы­ва­ют­ся их скорости. Зна­че­ние ско­ро­сти не может быть мень­ше 1 и боль­ше 300.Программа долж­на сна­ча­ла вы­ве­сти ми­ни­маль­ную скорость, затем YES или NO.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 74 69 63 96 | 63 YES |

**21. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

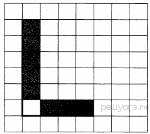
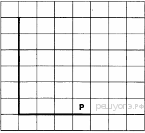
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся вертикальная стена. Длина стены неизвестна. От ниж­не­го конца стены впра­во отходит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной над пра­вым краем го­ри­зон­таль­ной стены. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные правее вер­ти­каль­ной стены, выше го­ри­зон­таль­ной стены и при­мы­ка­ю­щие к ним, кроме уг­ло­вой клетки. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

Конечное рас­по­ло­же­ние Робота может быть произвольным. Ал­го­ритм должен ре­шать задачу для про­из­воль­но­го размера поля и лю­бо­го допустимого рас­по­ло­же­ния стен внут­ри прямоугольного поля. При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите про­грам­му для ре­ше­ния следующей задачи. Ка­ме­ра наблюдения ре­ги­стри­ру­ет в ав­то­ма­ти­че­ском режиме ско­рость проезжающих мимо неё автомобилей, округ­ляя значения ско­ро­сти до целых чисел. Не­об­хо­ди­мо определить мак­си­маль­ную зарегистрированную ско­рость автомобиля. Если ско­рость хотя бы од­но­го автомобиля была не мень­ше 60 км/ч, вы­ве­ди­те «YES», иначе вы­ве­ди­те «N0».

Программа по­лу­ча­ет на вход число про­ехав­ших автомобилей N (1 =< N =< 30), затем ука­зы­ва­ют­ся их скорости. Зна­че­ние скорости не может быть мень­ше 1 и боль­ше 300. Про­грам­ма должна сна­ча­ла вывести сред­нюю скорость с точ­но­стью до од­но­го знака после запятой, затем «YES» или «N0».

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 74 69 63 96 | 75.5 YES |

**22. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

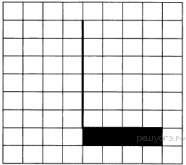
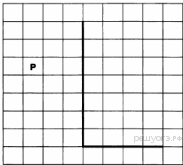
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся вер­ти­каль­ная стена. Длина стены неизвестна. От ниж­не­го конца стены впра­во от­хо­дит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной слева от вер­ти­каль­ной стены и выше го­ри­зон­таль­ной стены. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные ниже го­ри­зон­таль­ной стены и при­мы­ка­ю­щие к ней. Тре­бу­ет­ся за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите про­грам­му для ре­ше­ния сле­ду­ю­щей задачи. Ка­ме­ра на­блю­де­ния ре­ги­стри­ру­ет в ав­то­ма­ти­че­ском ре­жи­ме ско­рость про­ез­жа­ю­щих мимо неё автомобилей, округ­ляя зна­че­ния ско­ро­сти до целых чисел. Не­об­хо­ди­мо опре­де­лить сред­нюю за­ре­ги­стри­ро­ван­ную ско­рость всех автомобилей. Если не менее двух ав­то­мо­би­лей дви­га­лись со ско­ро­стью не боль­ше 40 км/ч, вы­ве­ди­те «YES», иначе вы­ве­ди­те «NO».

Программа по­лу­ча­ет на вход число про­ехав­ших ав­то­мо­би­лей N (1 =< N =< 30), затем ука­зы­ва­ют­ся их скорости. Зна­че­ние ско­ро­сти не может быть мень­ше 1 и боль­ше 300.Программа долж­на сна­ча­ла вы­ве­сти сред­нюю скорость, затем «YES» или «NO».

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 74 69 63 96 | 63 NO |

**23.**Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

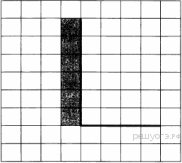
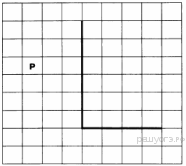
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся вер­ти­каль­ная стена. Длина стены неизвестна. От ниж­не­го конца стены впра­во от­хо­дит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной слева от вер­ти­каль­ной стены и выше го­ри­зон­таль­ной стены. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные слева от вер­ти­каль­ной стены и при­мы­ка­ю­щие к ней. Тре­бу­ет­ся за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите про­грам­му для ре­ше­ния сле­ду­ю­щей задачи. Ка­ме­ра на­блю­де­ния ре­ги­стри­ру­ет в ав­то­ма­ти­че­ском ре­жи­ме ско­рость про­ез­жа­ю­щих мимо неё автомобилей, округ­ляя зна­че­ния ско­ро­сти до целых чисел. Не­об­хо­ди­мо определить:

1) разность мак­си­маль­ной и ми­ни­маль­ной ско­ро­стей автомобилей;

2) количество автомобилей, ско­рость ко­то­рых не пре­вы­ша­ла 30 км/ч.

Программа по­лу­ча­ет на вход число про­ехав­ших ав­то­мо­би­лей N (1 < N < 30), затем ука­зы­ва­ют­ся их скорости. Зна­че­ние ско­ро­сти не может быть мень­ше 1 и боль­ше 300. Про­грам­ма долж­на сна­ча­ла вы­ве­сти раз­ность мак­си­маль­ной и ми­ни­маль­ной ско­ро­стей автомобилей, затем ко­ли­че­ство автомобилей, ско­рость ко­то­рых не пре­вы­ша­ла 30 км/ч.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 74 69 63 96 | 33 0 |

**24. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся прямоугольник, огра­ни­чен­ный стенами. Длины сто­рон пря­мо­уголь­ни­ка неизвестны. Робот на­хо­дит­ся внут­ри прямоугольника. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий верх­ние уг­ло­вые клетки. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти целых чисел опре­де­ля­ет ко­ли­че­ство нечётных чисел, крат­ных 3. Про­грам­ма по­лу­ча­ет на вход целые числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — признак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти два числа: длину по­сле­до­ва­тель­но­сти (завершающий 0 не учитывается) и ко­ли­че­ство нечётных чисел, крат­ных 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 6 15 180 0 | 4 1 |

**25. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

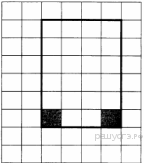
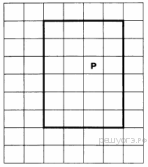
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся прямоугольник, огра­ни­чен­ный стенами. Длины сто­рон прямоугольника неизвестны. Робот на­хо­дит­ся внутри прямоугольника. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий нижние уг­ло­вые клетки. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

Конечное рас­по­ло­же­ние Робота может быть произвольным. Ал­го­ритм должен ре­шать задачу для про­из­воль­но­го размера поля и лю­бо­го допустимого рас­по­ло­же­ния стен внут­ри прямоугольного поля. При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти целых чисел опре­де­ля­ет их сумму и ко­ли­че­ство чётных чисел, крат­ных 5. Про­грам­ма получает на вход целые числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — при­знак окончания ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма должна вы­ве­сти два числа: сумму по­сле­до­ва­тель­но­сти и ко­ли­че­ство чётных чисел, крат­ных 5.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 60 15 0 | 79 1 |

**26. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

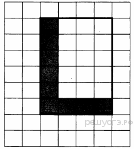
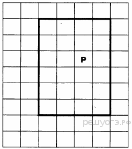
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся прямоугольник, огра­ни­чен­ный стенами. Длины сто­рон пря­мо­уголь­ни­ка неизвестны. Робот на­хо­дит­ся внут­ри прямоугольника. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий клетки прямоугольника, примыкающие к его нижней и верхней стенам. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти целых чисел опре­де­ля­ет их ко­ли­че­ство и сумму чётных чисел. Про­грам­ма по­лу­ча­ет на вход целые числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — признак окон­ча­ния ввода, не вхо­дит в последовательность).

Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти два числа: длину по­сле­до­ва­тель­но­сти и сумму чётных чисел.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 60 15 0 | 3 64 |

**27. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

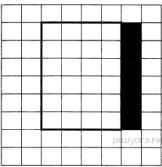
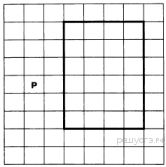
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся прямоугольник, огра­ни­чен­ный стенами. Длины сто­рон пря­мо­уголь­ни­ка неизвестны. Робот на­хо­дит­ся слева от прямоугольника. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий клет­ки сна­ру­жи прямоугольника, при­мы­ка­ю­щие сто­ро­ной к его пра­вой стене. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти целых чисел опре­де­ля­ет их сумму и под­счи­ты­ва­ет раз­ность ко­ли­че­ства по­ло­жи­тель­ных и от­ри­ца­тель­ных чисел последовательности. Про­грам­ма по­лу­ча­ет на вход целые числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность).

Количество чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти два числа: сумму чисел и раз­ность ко­ли­че­ства по­ло­жи­тель­ных и от­ри­ца­тель­ных чисел.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 150 −200 −1 0 | −51 −1 |

**28.**Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

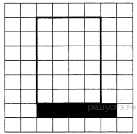
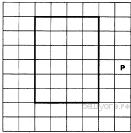
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся прямоугольник, огра­ни­чен­ный стенами. Длины сто­рон пря­мо­уголь­ни­ка неизвестны. Робот на­хо­дит­ся спра­ва от прямоугольника.На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий клет­ки сна­ру­жи прямоугольника, при­мы­ка­ю­щие сто­ро­ной к его ниж­ней стене, а также клетку, при­мы­ка­ю­щую к пра­во­му ниж­не­му углу. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти целых чисел опре­де­ля­ет их ко­ли­че­ство и под­счи­ты­ва­ет сумму по­ло­жи­тель­ных чётных чисел, не пре­вос­хо­дя­щих 256. Про­грам­ма по­лу­ча­ет на вход целые числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность).

Количество чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти два числа: длину по­сле­до­ва­тель­но­сти и сумму по­ло­жи­тель­ных чётных чисел, не пре­вос­хо­дя­щих 256.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| −20 6 1000 100 −200 0 | 5 106 |

**29. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

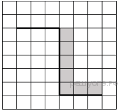
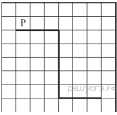
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из 3 по­сле­до­ва­тель­ных отрезков: вправо, вниз, вправо, все от­рез­ки неизвестной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной сверху ле­во­го конца пер­во­го отрезка. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные правее вто­ро­го отрезка и над третьим. Про­хо­ды должны остать­ся незакрашенными. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ко­неч­ное расположение Ро­бо­та может быть произвольным. Ал­го­ритм должен ре­шать задачу для лю­бо­го допустимого рас­по­ло­же­ния стен и лю­бо­го расположения и раз­ме­ра проходов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел опре­де­ля­ет количество всех чётных чисел, крат­ных 5. Про­грам­ма получает на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — при­знак окончания ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма должна вы­ве­сти одно число: ко­ли­че­ство всех чётных чисел, крат­ных 5.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 10 14 50 25 17 0 | 2 |

**30. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ←, впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

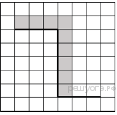
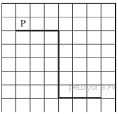
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из 3 по­сле­до­ва­тель­ных отрезков: вправо, вниз, вправо, все от­рез­ки не­из­вест­ной длины. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные над пер­вым от­рез­ком и спра­ва от второго. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­дов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство всех чётных чисел, крат­ных 9. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число: ко­ли­че­ство всех чётных чисел, крат­ных 9.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 27 24 18 22 36 0 | 2 |

**31. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

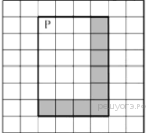
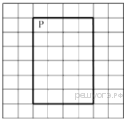
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся 4 стены, рас­по­ло­жен­ные в форме прямоугольника. Длины вер­ти­каль­ных и го­ри­зон­таль­ных стен неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной в левом верх­нем углу прямоугольника. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные с внут­рен­ней стороны пра­вой и ниж­ней стен. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ко­неч­ное расположение Ро­бо­та может быть произвольным. Ал­го­ритм должен ре­шать задачу для лю­бо­го допустимого рас­по­ло­же­ния стен и лю­бо­го расположения и раз­ме­ра проходов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел опре­де­ля­ет сумму всех чисел, крат­ных 7 и окан­чи­ва­ю­щих­ся на 2. Про­грам­ма получает на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — при­знак окончания ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма должна вы­ве­сти одно число: сумму всех чисел, крат­ных 7 и окан­чи­ва­ю­щих­ся на 2.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 112 24 42 49 22 0 | 154 |

**32. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

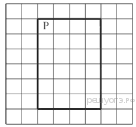
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся 4 стены, рас­по­ло­жен­ные в форме прямоугольника. Длины вер­ти­каль­ных и го­ри­зон­таль­ных стен неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной в левом верх­нем углу прямоугольника. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные с внут­рен­ней стороны верх­ней и ниж­ней стен. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ко­неч­ное расположение Ро­бо­та может быть произвольным. Ал­го­ритм должен ре­шать задачу для лю­бо­го допустимого рас­по­ло­же­ния стен и лю­бо­го расположения и раз­ме­ра проходов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел опре­де­ля­ет сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4. Про­грам­ма получает на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — признак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма должна вы­ве­сти одно число: сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 24 6 34 22 84 0 | 108 |

**33. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

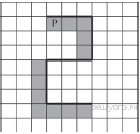
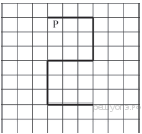
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, со­сто­я­щая из 5 по­сле­до­ва­тель­ных отрезков, рас­по­ло­жен­ных змейкой: вправо, вниз, влево, вниз, вправо, все от­рез­ки неизвестной длины. Робот на­хо­дит­ся в самой левой клет­ке непосредственно под верх­ней горизонтальной стеной. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные ниже пер­во­го и левее вто­ро­го отрезков стены и левее четвёртого и ниже пя­то­го отрезков стены. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ко­неч­ное расположение Ро­бо­та может быть произвольным. Ал­го­ритм должен ре­шать задачу для лю­бо­го допустимого рас­по­ло­же­ния стен и лю­бо­го расположения и раз­ме­ра проходов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел опре­де­ля­ет сумму всех чисел, крат­ных 8 и окан­чи­ва­ю­щих­ся на 6. Про­грам­ма получает на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — признак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30000. Про­грам­ма должна вы­ве­сти одно число: сумму всех на­ту­раль­ных чисел, крат­ных 8 и окан­чи­ва­ю­щих­ся на 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 16 24 56 22 12 0 | 72 |

**34.**Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

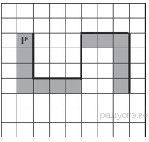
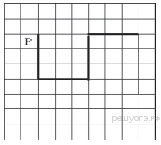
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, со­сто­я­щая из 5 по­сле­до­ва­тель­ных отрезков, рас­по­ло­жен­ных змейкой: вниз, вправо, вверх, вправо, вниз. Все от­рез­ки неизвестной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной слева от верх­не­го края пер­вой вертикальной стены. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные левее пер­во­го и ниже вто­ро­го отрезков стены и ниже четвёртого и левее пя­то­го отрезков стены. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ко­неч­ное расположение Ро­бо­та может быть произвольным. Ал­го­ритм должен ре­шать задачу для лю­бо­го допустимого рас­по­ло­же­ния стен и лю­бо­го расположения и раз­ме­ра проходов внут­ри стен. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Напишите программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел опре­де­ля­ет сумму всех чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 4. Про­грам­ма получает на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 — при­знак окончания ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30000. Про­грам­ма должна вы­ве­сти одно число: сумму всех на­ту­раль­ных чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 16 24 56 22 54 0 | 78 |

**35. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

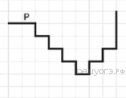
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся лестница. Сна­ча­ла лест­ни­ца спус­ка­ет­ся вниз слева направо, потом под­ни­ма­ет­ся вверх также слева направо. После подъ­ема лест­ни­ца пе­ре­хо­дит в вер­ти­каль­ную стену. Вы­со­та каж­дой ступени — 1 клетка, ширина — 1 клетка. Ко­ли­че­ство ступенек, ве­ду­щих вверх, и ко­ли­че­ство ступенек, ве­ду­щих вниз, неизвестно. Между спус­ком и подъ­емом ши­ри­на площадки — 1 клетка. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной в на­ча­ле спуска. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р») .

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но над лестницей, как по­ка­за­но на рисунке. Тре­бу­ет­ся за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2**Введите с кла­ви­а­ту­ры 5 по­ло­жи­тель­ных целых чисел. Вы­чис­ли­те сумму тех из них, ко­то­рые де­лят­ся на 4 и при этом за­кан­чи­ва­ют­ся на 6. Про­грам­ма долж­на вы­ве­сти одно число: сумму чисел, вве­ден­ных с клавиатуры, крат­ных 4 и окан­чи­ва­ю­щих­ся на 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 12 16 36 26 30 | 52 |

**36. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Исполнитель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может. У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды — это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся лестница. Сна­ча­ла лестница под­ни­ма­ет­ся вверх слева направо, потом опус­ка­ет­ся вниз также слева направо. Пра­вее спуска лест­ни­ца переходит в го­ри­зон­таль­ную стену. Вы­со­та каждой ступени — 1 клетка, ширина — 1 клетка. Ко­ли­че­ство ступенек, ве­ду­щих вверх, и ко­ли­че­ство ступенек, ве­ду­щих вниз, неизвестно. Между спус­ком и подъ­емом ширина площадки — 1 клетка. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной в на­ча­ле спуска. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р») .

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные непосредственно над лестницей. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

Конечное рас­по­ло­же­ние Робота может быть произвольным. Ал­го­ритм должен ре­шать задачу для про­из­воль­но­го размера поля и лю­бо­го допустимого рас­по­ло­же­ния стен внут­ри прямоугольного поля. При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го исполнителя или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те алгоритм в тек­сто­вом файле.

**20.2**Введите с кла­ви­а­ту­ры 8 по­ло­жи­тель­ных целых чисел. Определите, сколь­ко из них де­лят­ся на 3 и при этом за­кан­чи­ва­ют­ся на 4. Про­грам­ма должна вы­ве­сти одно число: ко­ли­че­ство чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 12 14 24 54 44 33 84 114 | 4 |

**37. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит команду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой закрашивается клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды проверки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вместе с усло­ви­ем **«если»**, име­ю­щим следующий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд проверки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клетки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва свободно то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать несколько ко­манд проверки условий, при­ме­няя логические связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния последовательности ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий следующий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

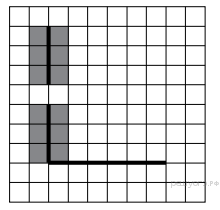
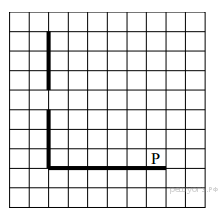
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать следующий алгоритм:

**нц пока спра­ва свободно**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле есть го­ри­зон­таль­ная и вер­ти­каль­ная стены. Пра­вый конец го­ри­зон­таль­ной стены соединён с ниж­ним концом вер­ти­каль­ной стены. Длины стен неизвестны. В го­ри­зон­таль­ной стене есть ровно один проход, точ­ное место про­хо­да и его ши­ри­на неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной непосредственно над го­ри­зон­таль­ной стеной у её пра­во­го конца. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные непосредственно левее и пра­вее вер­ти­каль­ной стены. Про­ход дол­жен остаться незакрашенным. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки(см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние алгоритма долж­но завершиться. Ко­неч­ное расположение Ро­бо­та может быть произвольным. Ал­го­ритм должен ре­шать задачу для лю­бо­го допустимого рас­по­ло­же­ния стен и лю­бо­го расположения и раз­ме­ра прохода внут­ри стены. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе. Со­хра­ни­те ал­го­ритм в тек­сто­вом файле.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ми­ни­маль­ное число, окан­чи­ва­ю­ще­е­ся на 4. Про­грам­ма получает на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти всегда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 4. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — ми­ни­маль­ное число, окан­чи­ва­ю­ще­е­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 3 24 14 34 | 14 |

**38. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

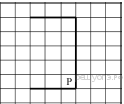
**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**



Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

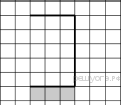
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, со­сто­я­щая из трёх по­сле­до­ва­тель­ных отрезков: вправо, вниз, влево. Все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной в ниж­нем углу, ко­то­рый об­ра­зу­ет­ся вто­рым и тре­тьим отрезком. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стены и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные ниже тре­тье­го отрезка. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для бес­ко­неч­но­го поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 8. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел заканчи- ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 8.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 23  48  12  18  34  0 | 66 |

**39. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

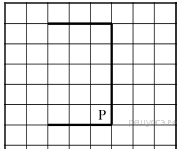
**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**



Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

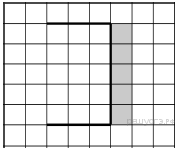
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, со­сто­я­щая из трёх по­сле­до­ва­тель­ных отрезков: вправо, вниз, влево. Все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной в ниж­нем углу, ко­то­рый об­ра­зу­ет­ся вто­рым и тре­тьим отрезком. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стены и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные правее второго отрезка. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для бес­ко­неч­но­го поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел заканчи- ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 54  28  72  34  24  0 | 78 |

**40. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

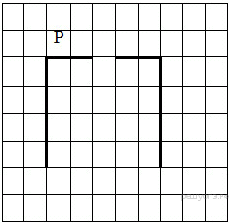
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

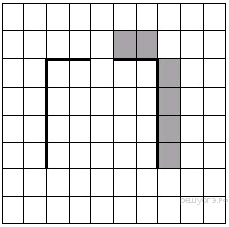
**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из двух вер­ти­каль­ных и со­еди­ня­ю­ще­го их го­ри­зон­таль­но­го от­рез­ков (отрезки стены рас­по­ло­же­ны "буквой П"). В го­ри­зон­таль­ном участ­ке есть ровно один проход, место и длина про­хо­да неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной над левым кон­цом го­ри­зон­таль­но­го от­рез­ка стены.

На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Робота

(Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные над го­ри­зон­таль­ным от­рез­ком стены спра­ва от прохода, и все клетки, рас­по­ло­жен­ные с внеш­ней сто­ро­ны от пра­во­го вер­ти­каль­но­го участ­ка стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться.

Алгоритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля, лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра прохода.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 2. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность).

Количество чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 2.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 12  24  22  72  11  0 | 84 |

**41. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

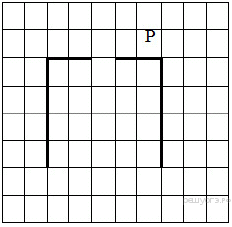
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

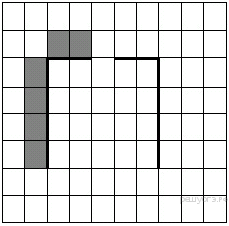
**Выполните задание.**



На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из двух вер­ти­каль­ных и со­еди­ня­ю­ще­го их го­ри­зон­таль­но­го от­рез­ков (отрезки стены рас­по­ло­же­ны "буквой П"). В го­ри­зон­таль­ном участ­ке есть ровно один проход, место и длина про­хо­да неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной над пра­вым кон­цом го­ри­зон­таль­но­го от­рез­ка стены.

На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Робота

(Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные над го­ри­зон­таль­ным от­рез­ком стены слева от прохода, и все клетки, рас­по­ло­жен­ные с внеш­ней сто­ро­ны от ле­во­го вер­ти­каль­но­го участ­ка стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию.

Например, для приведённого спра­ва ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться.

Алгоритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля, лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра прохода.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 4 и окан­чи­ва­ю­щих­ся на 8. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 – при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 4 и окан­чи­ва­ю­щих­ся на 8.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 8  22  16  28  18  0 | 36 |

**42. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

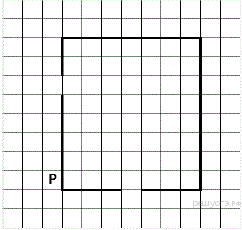
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

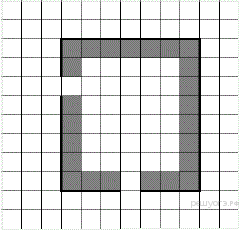
**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся че­ты­ре стены, соединённые между собой, ко­то­рые об­ра­зу­ют прямоугольник. Длины стен неизвестны. В левой вер­ти­каль­ной стене есть ровно один проход, в ниж­ней го­ри­зон­таль­ной стене

также есть ровно один проход. Про­ход не может при­мы­кать к углу прямоугольника. Точ­ные места про­хо­дов и ши­ри­на про­хо­дов неизвестны. Робот на­хо­дит­ся около ниж­не­го конца левой вер­ти­каль­ной стены, сна­ру­жи пря­мо­уголь­ни­ка и выше ниж­ней стены. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные вдоль стен пря­мо­уголь­ни­ка с внут­рен­ней стороны. Про­хо­ды долж­ны остать­ся незакрашенными. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).



При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го

расположения стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­да внут­ри стены. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ми­ни­маль­ное чётное число.

Программа по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся чётное число. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000.

Программа долж­на вы­ве­сти одно число — ми­ни­маль­ное чётное число.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  3  20  6  8 | 6 |

**43. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

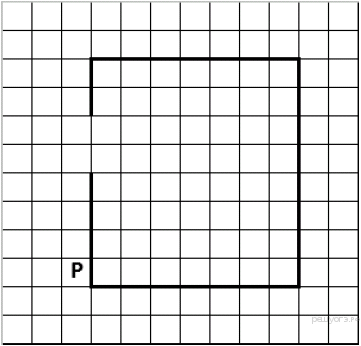
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

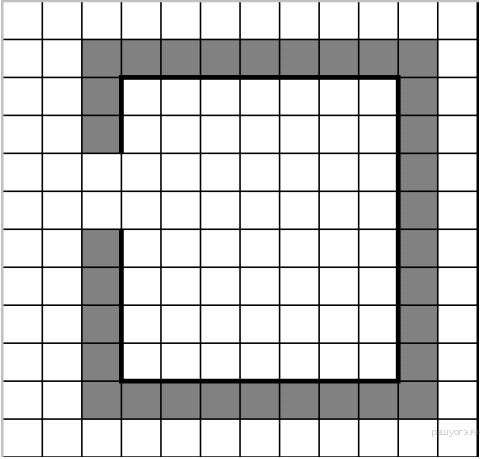
**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся че­ты­ре стены, соединённые между собой, ко­то­рые об­ра­зу­ют прямоугольник. Длины стен неизвестны. В левой вер­ти­каль­ной стене есть ровно один проход. Про­ход не может при­мы­кать к углу прямоугольника. Точ­ное место про­хо­да и ши­ри­на про­хо­да неизвестна. Робот на­хо­дит­ся около ниж­не­го конца левой вер­ти­каль­ной стены, сна­ру­жи пря­мо­уголь­ни­ка и выше ниж­ней стены. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные вдоль стен пря­мо­уголь­ни­ка с внеш­ней стороны. Про­ход должен остать­ся незакрашенным. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го

расположения стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­да внут­ри стены. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел опре­де­ля­ет максимальное число, окан­чи­ва­ю­ще­е­ся на 2. Программа по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти всегда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 2. Количество чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма должна вы­ве­сти одно число — мак­си­маль­ное число, окан­чи­ва­ю­ще­е­ся на 2.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  3  22  6  12 | 22 |

**44. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

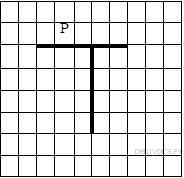
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

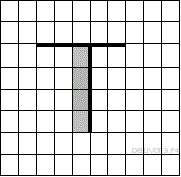
**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся две пер­пен­ди­ку­ляр­ные друг другу стены, рас­по­ло­жен­ные в виде буквы «Т», длины стен неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но над го­ри­зон­таль­ной стеной. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но с левой сто­ро­ны вер­ти­каль­ной стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого спра­ва ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го

расположения стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­да внут­ри стены. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 4 и окан­чи­ва­ю­щих­ся на 6. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 4 и окан­чи­ва­ю­щих­ся на 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 16  24  56  26  12  0 | 72 |

**45. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

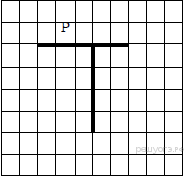
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

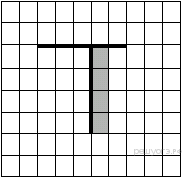
**вправо**

**кц**

**Выполните задание.**



На бес­ко­неч­ном поле име­ют­ся две пер­пен­ди­ку­ляр­ные друг другу стены, рас­по­ло­жен­ные в виде буквы «Т», длины стен неизвестны. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но над го­ри­зон­таль­ной стеной. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но с пра­вой сто­ро­ны вер­ти­каль­ной стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого спра­ва ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

 При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться, вы­пол­не­ние ал­го­рит­ма долж­но завершиться. Ко­неч­ное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен и лю­бо­го рас­по­ло­же­ния и раз­ме­ра про­хо­да внут­ри стены. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 9. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 9.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 16  29  56  9  39  0 | 48 |

**46. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

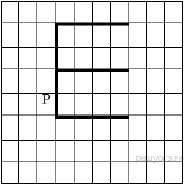
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

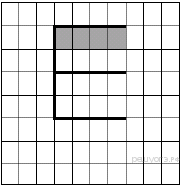
**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из од­но­го вер­ти­каль­но­го и трёх рав­ных го­ри­зон­таль­ных от­рез­ков (отрезки стены рас­по­ло­же­ны бук­вой «Е»). Все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но слева от ниж­не­го конца вер­ти­каль­но­го отрезка. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные под верх­ним го­ри­зон­таль­ным от­рез­ком стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого спра­ва ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

 Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 7 и окан­чи­ва­ю­щих­ся на 3. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 7 и окан­чи­ва­ю­щих­ся на 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 73  13  14  133  63  0 | 196 |

**47. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

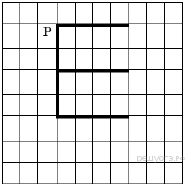
Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

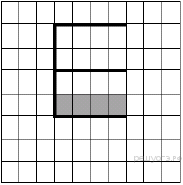
**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из од­но­го вер­ти­каль­но­го и трёх рав­ных го­ри­зон­таль­ных от­рез­ков (отрезки стены рас­по­ло­же­ны бук­вой «Е»). Все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но слева от верх­не­го конца вер­ти­каль­но­го отрезка. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные над ниж­ним го­ри­зон­таль­ным от­рез­ком стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого спра­ва ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

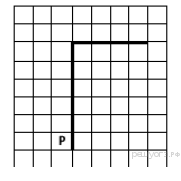
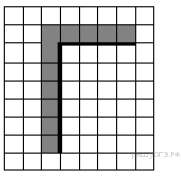
Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 7 и окан­чи­ва­ю­щих­ся на 1. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 7 и окан­чи­ва­ю­щих­ся на 1.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 21  14  31  28  91  0 | 112 |

**48. 20.1**

На бес­ко­неч­ном поле име­ет­ся вер­ти­каль­ная стена. Длина стены неизвестна. От верх­не­го конца стены впра­во от­хо­дит го­ри­зон­таль­ная стена также не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной слева от ниж­не­го края вер­ти­каль­ной стены.

На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные левее вер­ти­каль­ной стены и выше го­ри­зон­таль­ной стены и при­ле­га­ю­щие к ним. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм на­пи­ши­те в тек­сто­вом ре­дак­то­ре и со­хра­ни­те в тек­сто­вом файле. На­зва­ние файла и ка­та­лог для со­хра­не­ния Вам со­об­щат ор­га­ни­за­то­ры экзамена.

**20.2** Напишите программу, которая в последовательности целых чисел определяет количество чётных чисел, кратных 7. Программа получает на вход целые числа, количество введённых чисел неизвестно, последовательность чисел заканчивается числом 0 (0 — признак окончания ввода, не входит в последовательность). Количество чисел не превышает 1000. Введённые числа по модулю не превышают 30 000. Программа должна вывести одно число: количество чётных чисел, кратных 7.

Пример работы программы:

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| –32  14  17  0 | 1 |

**49. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

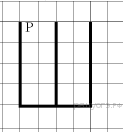
**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

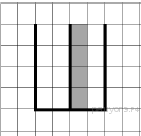
**нц пока спра­ва сво­бод­но**

**вправо**

**кц**



**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из од­но­го го­ри­зон­таль­но­го и трёх рав­ных вер­ти­каль­ных от­рез­ков (отрезки стены рас­по­ло­же­ны бук­вой «Ш»). Все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но спра­ва от верх­не­го конца ле­во­го вер­ти­каль­но­го отрезка. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р») .

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но спра­ва от вто­ро­го вер­ти­каль­но­го отрезка. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 – при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 14  24  36  84  66  0 | 108 |

**50. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

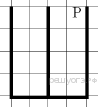
**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**



**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся стена, длины от­рез­ков стены неизвестны. Стена со­сто­ит из од­но­го го­ри­зон­таль­но­го и трёх рав­ных вер­ти­каль­ных от­рез­ков (отрезки стены рас­по­ло­же­ны бук­вой «Ш»). Все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но слева от верх­не­го конца прваого вер­ти­каль­но­го отрезка. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р») .

Напишите для Робота алгоритм, закрашивающий все клетки, расположенные непосредственно слева от второго вертикального отрезка. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).



Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 6. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 – при­знак окон­ча­ния ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти одно число: сумму всех чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 6.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 36  12  16  66  11  0 | 102 |

**51. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

https://inf-oge.sdamgia.ru/get_file?id=7440

**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся го­ри­зон­таль­ная стена. Длина стены неизвестна. Робот на­хо­дит­ся свер­ху от стены в левом её конце. На ри­сун­ке при­ве­де­но рас­по­ло­же­ние Ро­бо­та от­но­си­тель­но стены (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные выше стены на рас­сто­я­нии одной пу­стой клет­ки от стены, не­за­ви­си­мо от длины стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

https://inf-oge.sdamgia.ru/get_file?id=7441

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те про­грам­му для ре­ше­ния сле­ду­ю­щей задачи. Де­вя­ти­класс­ни­ки участ­во­ва­ли в вик­то­ри­не по математике. Не­об­хо­ди­мо было от­ве­тить на 20 вопросов. По­бе­ди­те­лем вик­то­ри­ны счи­та­ет­ся участник, пра­виль­но от­ве­тив­ший на наи­боль­шее ко­ли­че­ство вопросов. На сколь­ко во­про­сов по­бе­ди­тель от­ве­тил правильно? Если есть участ­ни­ки викторины, ко­то­рые не смог­ли дать пра­виль­ный ответ ни на один из вопросов, вы­ве­ди­те YES, иначе вы­ве­ди­те NO. Гарантируется, что есть участники, пра­виль­но от­ве­тив­шие хотя бы на один из вопросов. Про­грам­ма по­лу­ча­ет на вход число участ­ни­ков вик­то­ри­ны N (1 ≤ N ≤ 50), затем для каж­до­го участ­ни­ка вво­дит­ся ко­ли­че­ство вопросов, на ко­то­рые по­лу­чен пра­виль­ный ответ.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  15  12  0  17 | 17  YES |

**52. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

https://inf-oge.sdamgia.ru/get_file?id=7450

**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся го­ри­зон­таль­ная стена. Длина стены неизвестна. Робот на­хо­дит­ся свер­ху от стены в левом её конце. На ри­сун­ке при­ве­де­но рас­по­ло­же­ние Ро­бо­та от­но­си­тель­но стены (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные выше стены и при­ле­га­ю­щие к ней, не­за­ви­си­мо от раз­ме­ра стены и на­чаль­но­го рас­по­ло­же­ния Робота. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

https://inf-oge.sdamgia.ru/get_file?id=7449

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те про­грам­му для ре­ше­ния сле­ду­ю­щей задачи. Уче­ни­ки 4 клас­са вели днев­ни­ки на­блю­де­ния за по­го­дой и еже­днев­но за­пи­сы­ва­ли днев­ную температуру. Най­ди­те самую низ­кую тем­пе­ра­ту­ру за время наблюдения. Если тем­пе­ра­ту­ра опус­ка­лась ниже –15 градусов, вы­ве­ди­те YES, иначе вы­ве­ди­те NO. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство дней, в те­че­ние ко­то­рых про­во­ди­лось из­ме­ре­ние тем­пе­ра­ту­ры N (1 ≤ N ≤ 31), затем для каж­до­го дня вво­дит­ся температура.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  –5  12  –2  8 | –5  NO |

**53. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

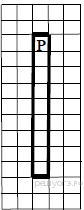
В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:



**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

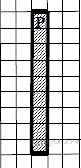
**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

Выполните задание.

Робот на­хо­дит­ся в верх­ней клет­ке уз­ко­го вер­ти­каль­но­го коридора. Ши­ри­на ко­ри­до­ра — одна клетка, длина ко­ри­до­ра может быть произвольной. Воз­мож­ный ва­ри­ант на­чаль­но­го рас­по­ло­же­ния Ро­бо­та приведён на ри­сун­ке (Робот обо­зна­чен бук­вой «Р»):



Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клет­ки внут­ри ко­ри­до­ра и воз­вра­ща­ю­щий Ро­бо­та в ис­ход­ную позицию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок). Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го ко­неч­но­го раз­ме­ра коридора. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 2. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число: ко­ли­че­ство чисел, крат­ных 3 и окан­чи­ва­ю­щих­ся на 2.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 4  12  25  12  9 | 2 |

**54. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

https://inf-oge.sdamgia.ru/get_file?id=7733

Выполните задание.

Робот на­хо­дит­ся в левой клет­ке уз­ко­го го­ри­зон­таль­но­го коридора. Ши­ри­на ко­ри­до­ра — одна клетка, длина ко­ри­до­ра может быть произвольной. Воз­мож­ный ва­ри­ант на­чаль­но­го рас­по­ло­же­ния Ро­бо­та приведён на ри­сун­ке (Робот обо­зна­чен бук­вой «Р»):

https://inf-oge.sdamgia.ru/get_file?id=7734

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клет­ки внут­ри ко­ри­до­ра и воз­вра­ща­ю­щий Ро­бо­та в ис­ход­ную позицию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок). Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го ко­неч­но­го раз­ме­ра коридора. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа по мо­ду­лю не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число: ко­ли­че­ство чисел, крат­ных 6 и окан­чи­ва­ю­щих­ся на 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 3  24  25  54 | 2 |

**55. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

https://inf-oge.sdamgia.ru/get_file?id=7762

Выполните задание.

На бес­ко­неч­ном поле име­ет­ся го­ри­зон­таль­ная стена. Длина стены неизвестна. Робот на­хо­дит­ся свер­ху от стены в левом ее конце. На ри­сун­ке при­ве­де­но рас­по­ло­же­ние ро­бо­та от­но­си­тель­но стены (робот обо­зна­чен бук­вой «Р»):

https://inf-oge.sdamgia.ru/get_file?id=7763

Напишите ал­го­ритм для робота, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные выше стены на рас­сто­я­нии одной пу­стой клет­ки от стены, не­за­ви­си­мо от длины стены. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие за­дан­но­му условию. Например, для приведённого выше ри­сун­ка робот дол­жен за­кра­сить сле­ду­ю­щие клетки.

Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет мак­си­маль­ное число, окан­чи­ва­ю­ще­е­ся на 3. Про­грам­ма по­лу­ча­ет на вход ко­ли­че­ство чисел в последовательности, а затем сами числа. В по­сле­до­ва­тель­но­сти все­гда име­ет­ся число, окан­чи­ва­ю­ще­е­ся на 3. Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число — мак­си­маль­ное число, оканчивающееся

на 3.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 3  13  23  3 | 23 |

**56. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

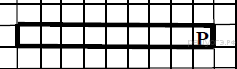
**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**



Выполните задание.

Робот на­хо­дит­ся в пра­вой клет­ке уз­ко­го го­ри­зон­таль­но­го коридора. Ши­ри­на ко­ри­до­ра — одна клетка, длина ко­ри­до­ра может быть произвольной. Воз­мож­ный ва­ри­ант на­чаль­но­го рас­по­ло­же­ния Ро­бо­та приведён на ри­сун­ке (Робот обо­зна­чен бук­вой «Р»):



Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клет­ки внут­ри ко­ри­до­ра и воз­вра­ща­ю­щий Ро­бо­та в ис­ход­ную позицию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок). Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го ко­неч­но­го раз­ме­ра коридора. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство трёхзначных чисел, крат­ных 4. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность).

Количество чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число: ко­ли­че­ство трёхзначных чисел, крат­ных 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 120  9  365  4  0 | 1 |

**57. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре команды — это команды-приказы:

**вверх вниз влево вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑ вниз ↓, влево ← , впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится.

Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре команды —  это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно  снизу свободно  слева свободно  спра­ва свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем **«если»**, име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* — одна из ко­манд про­вер­ки условия. *Последовательность команд* — это одна или не­сколь­ко любых команд-приказов. Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стенки, и за­кра­ши­ва­ния клет­ки можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки **и, или, не,** например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл **«пока»**, име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

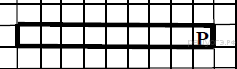
**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

**вправо**

**кц**



Выполните задание.

Робот на­хо­дит­ся в пра­вой клет­ке уз­ко­го го­ри­зон­таль­но­го коридора. Ши­ри­на ко­ри­до­ра — одна клетка, длина ко­ри­до­ра может быть произвольной. Воз­мож­ный ва­ри­ант на­чаль­но­го рас­по­ло­же­ния Ро­бо­та приведён на ри­сун­ке (Робот обо­зна­чен бук­вой «Р»):



Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клет­ки внут­ри ко­ри­до­ра и воз­вра­ща­ю­щий Ро­бо­та в ис­ход­ную позицию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок). Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го ко­неч­но­го раз­ме­ра коридора. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться. Ал­го­ритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел опре­де­ля­ет ко­ли­че­ство трёхзначных чисел, крат­ных 4. Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 — при­знак окон­ча­ния ввода, не вхо­дит в последовательность).

Количество чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000. Про­грам­ма долж­на вы­ве­сти одно число: ко­ли­че­ство трёхзначных чисел, крат­ных 4.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 120  9  365  4  0 | 1 |

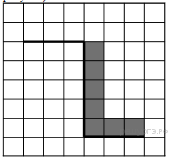
**58. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре ко­ман­ды – это команды-приказы:

**вверх   вниз   влево   вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑, вниз ↓, влево ←, впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре ко­ман­ды – это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно   снизу свободно   слева свободно   справа свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем «**eсли**», име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* – одна из ко­манд про­вер­ки условия.

*Последовательность команд* — это одна или не­сколь­ко любых команд-приказов.

Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стен­ки и за­кра­ши­ва­ния клетки, можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки и, или, не, например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл

«**пока**», име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

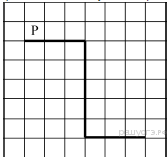
**нц пока спра­ва сво­бод­но**

**вправо**

**кц**

**Выполните задание.**

На бес­ко­неч­ном поле име­ет­ся стена. Стена со­сто­ит из трёх по­сле­до­ва­тель­ных отрезков: вправо, вниз, вправо, все от­рез­ки не­из­вест­ной длины. Робот на­хо­дит­ся в клетке, рас­по­ло­жен­ной не­по­сред­ствен­но свер­ху ле­во­го конца

первого отрезка. На ри­сун­ке ука­зан один из воз­мож­ных спо­со­бов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен бук­вой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клетки, рас­по­ло­жен­ные не­по­сред­ствен­но пра­вее вто­ро­го от­рез­ка и над третьим. Робот дол­жен за­кра­сить толь­ко клетки, удо­вле­тво­ря­ю­щие дан­но­му условию. Например, для приведённого выше ри­сун­ка Робот дол­жен за­кра­сить сле­ду­ю­щие клет­ки (см. рисунок).

 Конечное рас­по­ло­же­ние Ро­бо­та может быть произвольным. Ал­го­ритм дол­жен ре­шать за­да­чу для про­из­воль­но­го раз­ме­ра поля и лю­бо­го до­пу­сти­мо­го рас­по­ло­же­ния стен внут­ри пря­мо­уголь­но­го поля. При ис­пол­не­нии ал­го­рит­ма Робот не дол­жен разрушиться.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти на­ту­раль­ных чисел на­хо­дит сред­нее ариф­ме­ти­че­ское чисел, крат­ных 8, или сообщает, что таких чисел нет (выводит «NO»). Про­грам­ма по­лу­ча­ет на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно,последовательность чисел за­кан­чи­ва­ет­ся чис­лом 0 (0 – при­знак окон­ча­ния ввода, не вхо­дит в последовательность).

Количество чисел не пре­вы­ша­ет 100. Введённые числа не пре­вы­ша­ют 300. Про­грам­ма долж­на вы­ве­сти сред­нее ариф­ме­ти­че­ское чисел, крат­ных 8, или вы­ве­сти «NO», если таких чисел нет. Зна­че­ние вы­во­дить с точ­но­стью до десятых.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 8  122  64  16  0 | 29,3 |
| 111  1  0 | NO |

**59. Выберите ОДНО из пред­ло­жен­ных ниже заданий: 20.1 или 20.2.**

**20.1** Ис­пол­ни­тель Робот умеет пе­ре­ме­щать­ся по лабиринту, на­чер­чен­но­му на плоскости, раз­би­той на клетки. Между со­сед­ни­ми (по сторонам) клет­ка­ми может сто­ять стена, через ко­то­рую Робот прой­ти не может.

У Ро­бо­та есть де­вять команд. Че­ты­ре ко­ман­ды – это команды-приказы:

**вверх   вниз   влево   вправо**

При вы­пол­не­нии любой из этих ко­манд Робот пе­ре­ме­ща­ет­ся на одну клет­ку соответственно: вверх ↑, вниз ↓, влево ←, впра­во →. Если Робот по­лу­чит ко­ман­ду пе­ре­дви­же­ния сквозь стену, то он разрушится. Также у Ро­бо­та есть ко­ман­да **закрасить**, при ко­то­рой за­кра­ши­ва­ет­ся клетка, в ко­то­рой Робот на­хо­дит­ся в на­сто­я­щий момент.

Ещё че­ты­ре ко­ман­ды – это ко­ман­ды про­вер­ки условий. Эти ко­ман­ды проверяют, сво­бо­ден ли путь для Ро­бо­та в каж­дом из четырёх воз­мож­ных направлений:

**сверху свободно   снизу свободно   слева свободно   справа свободно**

Эти ко­ман­ды можно ис­поль­зо­вать вме­сте с усло­ви­ем «**eсли**», име­ю­щим сле­ду­ю­щий вид:

**если** *условие* **то**

*последовательность команд*

**все**

Здесь *условие* – одна из ко­манд про­вер­ки условия.

*Последовательность команд* — это одна или не­сколь­ко любых команд-приказов.

Например, для пе­ре­дви­же­ния на одну клет­ку вправо, если спра­ва нет стен­ки и за­кра­ши­ва­ния клетки, можно ис­поль­зо­вать такой алгоритм:

**если спра­ва сво­бод­но то**

**вправо**

**закрасить**

**все**

В одном усло­вии можно ис­поль­зо­вать не­сколь­ко ко­манд про­вер­ки условий, при­ме­няя ло­ги­че­ские связ­ки и, или, не, например:

**если (справа свободно) и (не снизу свободно) то**

**вправо**

**все**

Для по­вто­ре­ния по­сле­до­ва­тель­но­сти ко­манд можно ис­поль­зо­вать цикл

«**пока**», име­ю­щий сле­ду­ю­щий вид:

**нц пока** *условие*

*последовательность команд*

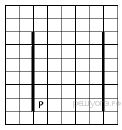
**кц**

Например, для дви­же­ния вправо, пока это возможно, можно ис­поль­зо­вать сле­ду­ю­щий алгоритм:

**нц пока спра­ва сво­бод­но**

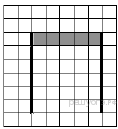
**вправо**

**кц**



**Выполните задание.**

На бес­ко­неч­ном поле име­ют­ся две вер­ти­каль­ные стены оди­на­ко­вой длины, рас­по­ло­жен­ные точно одна на­про­тив другой. Длина стен неизвестна. Рас­сто­я­ние между сте­на­ми неизвестно. Робот на­хо­дит­ся справа от пер­вой стены в клетке, рас­по­ло­жен­ной у её ниж­не­го края. На ри­сун­ке указан один из воз­мож­ных способов рас­по­ло­же­ния стен и Ро­бо­та (Робот обо­зна­чен буквой «Р»).

Напишите для Ро­бо­та алгоритм, за­кра­ши­ва­ю­щий все клет­ки самого верх­не­го ряда, рас­по­ло­жен­ные между стенами. Робот дол­жен закрасить толь­ко клетки, удо­вле­тво­ря­ю­щие данному условию. Например, для приведённого выше ри­сун­ка Робот дол­жен закрасить сле­ду­ю­щие клетки (см. рисунок).

 Конечное рас­по­ло­же­ние Робота может быть произвольным. Ал­го­ритм должен ре­шать задачу для про­из­воль­но­го размера поля и лю­бо­го допустимого рас­по­ло­же­ния стен внут­ри прямоугольного поля. При ис­пол­не­нии алгоритма Робот не дол­жен разрушиться.

Алгоритм может быть вы­пол­нен в среде фор­маль­но­го ис­пол­ни­те­ля или за­пи­сан в тек­сто­вом редакторе.

**20.2** На­пи­ши­те программу, ко­то­рая в по­сле­до­ва­тель­но­сти натуральных чисел вы­чис­ля­ет сумму всех дву­знач­ных чисел, крат­ных 8. Про­грам­ма получает на вход на­ту­раль­ные числа, ко­ли­че­ство введённых чисел неизвестно, по­сле­до­ва­тель­ность чисел за­кан­чи­ва­ет­ся числом 0 (0 – при­знак окончания ввода, не вхо­дит в последовательность). Ко­ли­че­ство чисел не пре­вы­ша­ет 1000. Введённые числа не пре­вы­ша­ют 30 000.

Программа долж­на вывести одно число: сумму всех дву­знач­ных чисел, крат­ных 8.

**Пример ра­бо­ты программы:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| 17  16  32  160  0 | 48 |