Игра «Змейка». Отображение движения змейки в одном направлении (одномерные массивы).

Темы: Инструкции цикла (while, for), перечисления, директивы препроцессора, одномерные массивы

В этом задании, как и в задании «Задание (игра Змейка) 2 (змейка на экране)», не требуется обрабатывать пользовательский ввод. На этот раз мы также будем изображать движущуюся змейку, но теперь она должна двигаться в пределах ограниченного поля.

**Замечание**

Текущая версия программы все еще будет упрощенной.

* Игровое поле будет состоять только из одной строки.
* Обработка нажатия клавиш производиться не будет
* Весь рабочий код пока будет располагаться в функции main.
* Пока змейка будет двигаться в бесконечном цикле. Прервать работу мы можем только, закрыв приложение насильственно.

В отличие от работы «Задание (игра Змейка)\_2 (змейка на экране)» здесь мы будем применять одномерные массивы для хранения значений игрового поля и координат змейки на этом игровом поле. Применение массивов сделает программу более простой по сравнению с приемом, который применялся в предыдущей работе (вывод ряда пробелов перед выводом символа змейки).

(Если бы нам пришлось добавить возможность изменения длины змейки, то код сильно усложнился бы)

Однако, в текущей работе мы будем применять одномерные массивы. Ограничения, которые накладываются на программу связаны с тем, что пока мы не знакомы с понятием двумерного массива, поэтому змейка может двигаться только в одном направлении.

**Задание1**

1. Добавьте в проект файлы, которые были созданы в предыдущем задании:

* snake.cpp и snake.h, которые будут содержать информацию, относящуюся к понятию "Змейка"
* game.cpp и game.h, которые будут содержать информацию, относящуюся к понятию "Игра".
* Заголовочный файл const.h, который будет содержать общие константные данные
* Файл main.cpp, который будет содержать все управляющие инструкции

1. Добавьте в проект новые файлы:

field.cpp и field.h, которые будут содержать информацию, относящуюся к понятию "Поле игры".

1. В файл const.h добавьте константы для отображения поля игры и границы поля, а также максимальное значение длины змейки

## Глобальные данные

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | переменная | Модуль | Примечания |
| Текущий размер змейки | snake\_size | snake | >0  Включает голову+размер хвоста  (если 1- то только голова) |
| Таймаут задержки между шагами игры | timeout | game | >100 (ms) |
| Длина игрового поля (включает граничные элементы) | columns | const.h | const, >0 |
| Символ для головы змейки | head\_symbol | const.h | const |
| Символ для хвоста змейки | tail\_symbol | const.h | const |
| Символ для заполнения поля игры | field\_symbol | const.h | Const |
| Символ для границы поля игры | border\_symbol | const.h | Const |
| Максимальный размер массива змейки | L | const.h | Const, не должна быть > длины игрового поля-3.(2 граничных элемента +1 элемент еды) |
| Массив  координат змейки по горизонтали (X) | snake\_x[L] | snake | >=0 |
| Массив игрового поля | field [columns] | field | >=0, пока одномерный |

Желтым отмечены новые данные (по сравнению с предыдущим проектом)

**Важное пояснение:**

Массив координат змейки по горизонтали (X) должен содержать **индексы поля игры**, в которых змейка присутствует

**Задание 2**

Так как работать пока умеем только с одномерными массивами, то сохранить сможем только одну строчку игрового поля.

В этой версии игры змейка будет отображаться вместе с полем, поэтому надо сохранять где-то координаты змейки (пока только головы), которые она занимает в игровом поле. Для сохранения этих координат будем использовать одномерный массив. В этом массиве координата головы будет храниться в элементе с индексом [0].

Имеет смысл при определении массивов проинициализировать их нулевыми значениями.

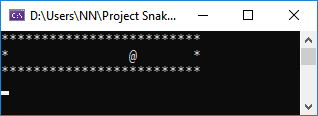
Определите и проинициализируйте в соответствующих модулях нужные переменные и константы:

* Максимальный размер массива змейки
* таймаут задержки между шагами игры
* Массив координат X змейки (одномерный).
* Массив игрового поля (одномерный).

**Задание 3**

Начать игру следует с вывода на экран игрового поля, на котором изображается змейка (пока только ее голова) и приглашения начать игру.

* Задайте значение длины змейки, равное 1.
* В массиве поля игры будем хранить и граничные элементы поля (левую и правую границы), поэтому координата X головы не может принимать значения 0 и column-1



Алгоритм этого фрагмента может быть таким:

1. «Очистка» всего игрового поля
2. Задание начального положения змейки
3. “Установка” змейки в поле игры
4. Вывод на экран поля игры.
5. Вывод приглашения к игре.

Замечания:

1. **Самое хитрое!**

Самым необычным в этой реализации будет “установка” символов змейки (пока только головы) в игровом поле. Необычным является то, что в качестве индекса массива field будем использовать **значение элемента другого массива** (змейки).

Массив координат змейки является индексным массивом.

Т.е. в те координаты **поля**, где стоит змейка, надо присвоить символы **змейки**, и потом печатать поле (c “установленной” змейкой):

например, snake\_x[0]=3, это означает, что голова находится в позиции 3 поля (координата по горизонтали, сейчас пока у нас поле одномерное и храним только значения по горизонтали), тогда  
field[snake\_x[0]]==field[3] и можно туда символ головы положить.

Обратите внимание, что в качестве индекса массива field используется **значение** индексного массива snake\_x[ ].



Если данные принимают такие значения:

|  |  |  |
| --- | --- | --- |
| Переменная/константа | Модуль | Значение |
| snake\_size | snake | 3 |
| field[columns] | field |  |
| columns |  | 6 |

field**[**snake**[**0**]]**= ‘@’; // field**[3]**= ‘@’

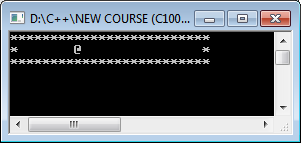
1. Для вывода приглашения начать игру надо приостановить выполнение программы пока пользователь не нажмет какую-то клавишу. А для этого можно использовать функцию \_getch (), которая считывает скан-код клавиши (как мы делали в первой работе по змейке). Сам код нас не интересует, поэтому обрабатывать его не надо.

Этот прием можно использовать для того, чтобы управлять началом игры, т.е. стартовать только после нажатия какой-либо клавиши. При желании можно выполнить обработку клавиатурного ввода и стартовать при нажатии определенной клавиши.

**Задание 4**

Будем изображать движущуюся змейку (состоящую только из головы). Раз змейка должна двигаться, то потребуется инструкция цикла.

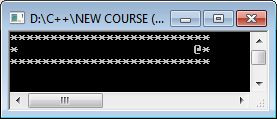
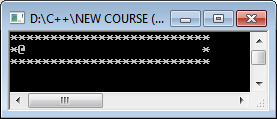
В текущей версии программы выполнение будет происходить «бесконечно», т.е. пока пользователь не закроет окно программы.



Важно, что продвигаться змейка может только тогда, когда «есть, куда двигаться», т.е. следующая позиция игрового поля - пробел.

В противном случае:

* Если голова змейки упирается в стенку, то движение змейки идет по кольцу, т.е. она появляется с другого (противоположного) края поля.
* Для проверки того, что змейка уперлась в правый край поля потребуется знать размер игрового поля.

Для того, чтобы получить эффект «анимации», надо «стирать» (пробелами) старые позиции змейки на поле игры. Это можно сделать разными способами («стирать» все поле игры или только позиции, нанимаемые змейкой).

Алгоритм одного шага игры можно представить так:

1. Очистка экрана
2. «Очистка» всего массива игрового поля или только старых позиций змейки в поле игры (на Ваш выбор)
3. Изменение положения змейки (продвижение в нужном направлении). Пока двигаемся только вправо.
4. Если змейка уперлась в граничный элемент, то она должна появиться с другого (противоположного) края поля.
5. Установка новых позиций змейки в поле игры
6. Вывод поля игры на экран .
7. Временная задержка (Sleep).

**Задание 5**

В этом задании будем изображать не только голову, но и хвост змейки, поэтому в массиве змейки будем хранить кроме координат головы, также координаты хвоста змейки.

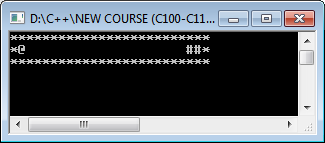
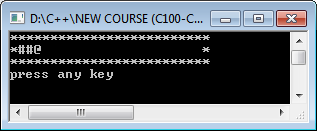


При передвижении должно происходить следующее:

1 способ

* Над элементами массива координат хвоста выполняется сдвиг вправо на 1 позицию.
* Координата головы змейки увеличивается на 1





**Совет:**

**Очень полезно** для дальнейшей работы комментировать фрагменты программы.