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)», мы также будем изображать движущуюся змейку, но теперь она должна двигаться в пределах ограниченного поля, причем, только по горизонтали.

**Замечание**

Текущая версия программы все еще будет упрощенной.

* Игровое поле будет состоять только из одной строки.
* Для завершения работы программы будем использовать клавишу **ESC**.
* Клавиши изменения направления движения змейки пока использовать не будем.   
  Весь рабочий код пока будет располагаться в функции main.

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 символов змейки).

(Если бы нам пришлось в предыдущей работе добавлять возможность изменения длины змейки, то код сильно усложнился бы).

В текущей работе мы будем применять **одномерные** массивы. Ограничения, которые накладываются на реализацию приложения связаны с тем, что пока мы не знакомы с понятием двумерного массива, поэтому змейка сможет двигаться только в горизонтальном направлении.

**Задание1**

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

## Данные модуля const

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип |  |
| Длина игрового поля по горизонтали (включает граничные элементы) | columns |  |  |
| Символ для хвоста змейки | tail\_symbol |  |  |
| Символ для головы змейки | head\_symbol |  |  |
| Символ для заполнения поля игры | field\_symbol |  |  |
| Символ для границы поля игры | border\_symbol |  |  |
| Максимальный размер массива змейки | L |  | const, не должна быть > длины игрового поля-2 |
| Перечисление «направление движения» | DIRECTION |  |  |
| Константы для обработки клавиатурного ввода (перечисление из “Задание (игра Змейка)\_1 (клавиатурный ввод)” ) | KYES |  |  |

Желтым отмечены новые данные (по сравнению с предыдущим проектом)

### Управление змейкой

В текущей версии проекта змейка будет двигаться в одномерном поле игры. Важно, что в том случае, когда используется одномерный массив в качестве поля, изменять направление движения змейки во время работы программы будет **нельзя**. Но направление движения можно задавать (единожды) в начале работы.   
Для того, чтобы задавать направление движения змейки удобно использовать перечисление, например, DIRECTION.

## 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) |
| Направление движения змейки | direction | | snake | | В качестве типа для переменной  удобно использовать перечисление, например, DIRECTION |
| Массив  координат змейки по горизонтали (X) | snake\_x[L] | | snake | | >=0 |
| Массив игрового поля | field [columns] | | field | | >=0, пока одномерный |
| Признак продолжения игры | game\_on | game | |  | |

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 верии 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.

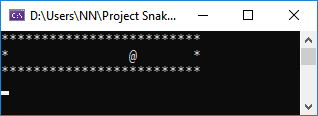
Определите и проинициализируйте в соответствующих модулях нужные переменные и константы:

* Максимальный размер массива змейки
* Перечисление KEYS для обработки клавиатурного ввода
* таймаут задержки между шагами игры
* перечисление DIRECTION, содержащее именованные константы LEFT и RIGHT.
* Переменную типа DIRECTION , определяющую направление движения змейки
* Массив координат X змейки (одномерный).
* Массив игрового поля (одномерный).
* Признак продолжения игры

**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 подготовительного фрагмента программы 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).

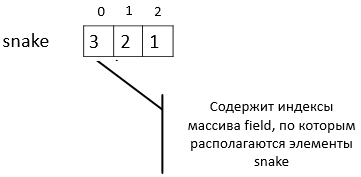
Массив координат змейки является индексным массивом.

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. Цикл завершает работу, когда пользователем будет нажата клавиша ESC. При этом признак продолжения игры **game\_on** устанавливается в false.

Graphical user interface, text, application

Description automatically generated

Важно, что продвигаться змейка может только тогда, когда «есть, куда двигаться», т.е. следующая позиция игрового поля пустая (занята символом-заполнителем поля).

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.

Кроме того, в зависимости от заданного направления движения (влево/вправо) будет изменяться текущая координата головы змейки.

Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated

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, занимаемые змейкой).

## Совет для упрощения отладки приложения

Передвижение змейки можно осуществлять в двух режимах:

* автоматическом
* пошаговом.

В предыдущем проекте при анимации движения змейки применялся автоматический режим. С помощью функции **Sleep() в**ыполнялась временная задержка между шагами алгоритма.

Пошаговое управление змейкой очень удобно для отладки алгоритмов передвижения змейки. После того, как будет отлажен алгоритм работы в пошаговом режиме, можно запускать автоматическое выполнение программы.

Алгоритм одного шага игры можно представить так:

1. Очистка позиций змейки на экране
2. Изменение положения змейки (продвижение в нужном направлении).
3. Проверка столкновения с границей поля
4. Установка новых позиций змейки в поле игры
5. Вывод поля игры на экран.
6. Проверка на конец игры
7. **«Задержка» выполнения игры**

### Пошаговый режим

Если используется пошаговый режим, надо приостановить выполнение программы **вручную**. Это можно сделать с помощью функции \_getch(). Программа будет приостановлена, пока пользователь не нажмет любую клавишу.

### Автоматический режим

Иначе, если используется автоматический режим, то надо приостановить выполнение программы на заданный промежуток времени. Это можно сделать с помощью функции **Sleep**().

Но в то же время нам хотелось бы обрабатывать клавиатурный ввод (нажатие клавиши **ESC**). В случае автоматического выполнения игры будем осуществлять **проверку факта нажатия клавиши клавиатуры** с помощью функции стандартной библиотеки **\_kbhit ()**.

Описание функции:

int \_ **kbhit** (void );

Функция \_**kbhit** проверяет консоль на предмет недавнего нажатия клавиши. Если функция возвращает ненулевое значение, код нажатой клавиши ожидает в буфере. Программа может затем вызвать \_getch для получения кода клавиши. Если ни одна клавиша не была нажата, то возвращается значение 0.

Эта функция объявлена в файле <conio.h>, поэтому требуется его подключить директивой препроцессора #include.

Последовательность действий

* Вызов \_kbhit()
* если ни одна клавиша не нажата, то переходим к следующей итерации (continue)
* если клавиша нажата, то проводим обработку нажатия

### Удобно использовать директивы условной трансляции

Для того, чтобы можно было переключаться от одного режима работы к другому без корректировки кода рекомендуем использовать директивы условной трансляции препроцессора.

#ifdef AUTOMATIC // автоматический режим

* Временная задержка (вызов функции Sleep())
* Вызов \_kbhit()
* если нажата клавиша ESC, то сбрасываем флаг продолжения игры
* иначе, ничего не делаем (пока!)

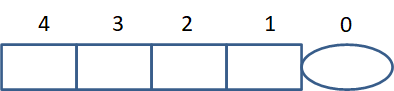
#else //пошаговый режим

* если нажата клавиша ESC, то сбрасываем флаг продолжения игры
* иначе, ничего не делаем (пока!)

#endif

**Задание 5**

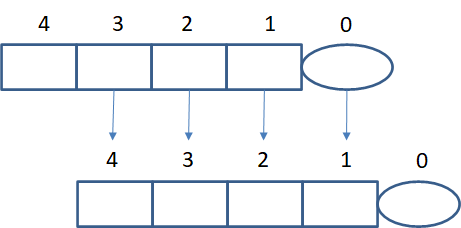
В этом задании будем изображать не только голову, но и хвост змейки, поэтому в массиве змейки будем хранить кроме координат головы, также координаты хвоста змейки.



Замечание: на картинке сверху указаны номера элементов массива змейки

При передвижении должно происходить следующее:

* Над элементами хвоста выполняется сдвиг вправо на 1 позицию.



snake\_x[3]  snake\_x[4]

snake\_x[2]  snake\_x[3]

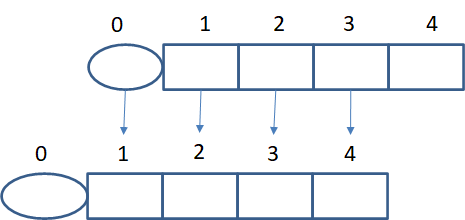
snake\_x[1]  snake\_x[2]

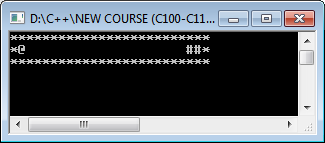
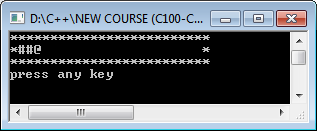
snake\_x[0]  snake\_x[1]

* Координата головы (snake\_x[0]) змейки увеличивается на 1, если движение вправо, и уменьшается на 1, если движение влево.

Как видно из предоставленных картинок, алгоритм «передвижения» хвоста НЕ зависит от направления движения змейки.

Chart, box and whisker chart

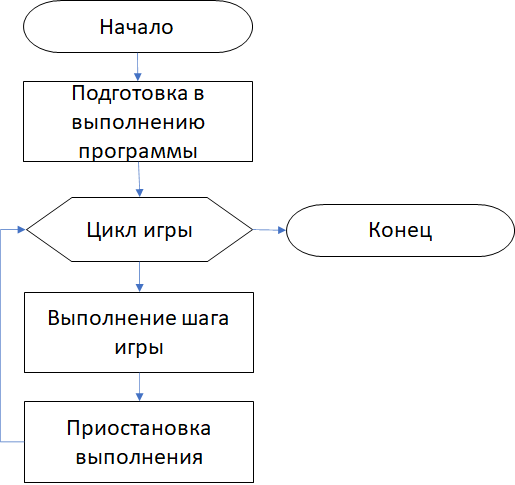
Description automatically generated 



**Совет:**

**Очень полезно** для дальнейшей работы комментировать фрагменты программы. Это впоследствии поможет грамотно разбить код на функции.

### Укрупненно алгоритм программы может выглядеть следующим образом



**Задание 6**

Проверьте, что программа работает корректно:

* змейка умеет передвигаться в горизонтальном направлении (влево, вправо)
* змейка умеет корректно преодолевать препятствия
* при нажатии клавиши ESC происходит завершение работы программы