Игра «Змейка». Отображение движения змейки в 2-мерном встроенном массиве

Темы: Двумерные встроенные массивы, функции с параметрами.

В этом задании мы также будем изображать движущуюся змейку, которая может съедать «еду», но в отличие от предыдущих проектов движение змейки будет осуществляться в разных направлениях и им можно будет управлять. Текущий проект будет разрабатываться на основе предыдущего.



**Задание 1**

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 константные данные
* Файл main.cpp, который будет содержать все управляющие инструкции
* field.cpp и field.h, которые будут содержать информацию, относящуюся к понятию "Поле игры".
* food.cpp и food.h, которые будут содержать информацию, относящуюся к понятию "Еда".

## Игровое поле

Удобно для моделирования игрового поля использовать встроенный двумерный массив типа **char**. Массив будет хранить символы, отображающие: границу, змейку и еду.

## Управление змейкой

В текущей версии проекта змейка сможет двигаться в двумерном поле игры. Для того, чтобы задавать направление движения змейки удобно использовать перечисление, например, DIRECTION.

**Задание 2**

**Важно (НОВЫЕ данные)!**

Так как поле игры теперь будет двумерным, то потребуется ввести новые данные:

* Количество **строк** игрового поля (константа)
* Координата расположения еды по вертикали
* Массив координат змейки в поле игры (**по** **вертикали**).
* Константы для обработки клавиатурного ввода
* Перечисление DIRECTION
* Переменная, хранящая текущее значение направления движения

Добавьте новые данные в программу

## Данные модуля const

|  |  |  |
| --- | --- | --- |
| Описание | Идентификатор | тип |
| Длина игрового поля по горизонтали | columns |  |
| Длина игрового поля по вертикали | rows |  |
| Максимальный размер массива змейки | L |  |
| Символ для хвоста змейки | tail\_symbol |  |
| Символ для головы змейки | head\_symbol |  |
| Символ еды | food\_symbol |  |
| Символ поля | field\_symbol |  |
| Символ границы | border\_symbol |  |
| Перечисление «направление движения» | DIRECTION |  |
| Константы для обработки клавиатурного ввода |  |  |

## Данные (переменные) модуля main

|  |  |  |
| --- | --- | --- |
| Описание | Идентификатор | Тип |
| Текущий размер змейки |  |  |
| Массив координат змейки по горизонтали (X) |  |  |
| Массив координат змейки по вертикали (X) |  |  |
| Массив игрового поля |  |  |
| Признак, что еда установлена |  |  |
| Координата еды по горизонтали (X) |  |  |
| Координата еды по вертикали (Y) |  |  |
| Таймаут задержки между шагами игры |  |  |
| Признак продолжения игры |  |  |
| Направление движения змейки |  |  |

## Первый шаг: частичная корректировка функций

**Задание 3**

Поскольку теперь игровое поле будет двумерным, то придется изменить объявления и реализации функций, которые используют массив поля.

Измените объявления и реализации функций, в которых используется поле игры. Алгоритмы функций меняться не должны, но так как змейка должна двигаться в двумерном поле, то реализации функций изменятся.

Все изменения будем вносить **постепенно**, чтобы не сломать все, что было сделано.

### Функции модуля field

1. На первом этапе внесем изменения в функции модуля **field. О**дномерный массив поля требуется заменить на двумерный.

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Параметры |
| Вывод поля на экран |  | Массив поля, [возможно что-то еще] |
| Инициализация поля |  | Массив поля, [возможно что-то еще] |
| Очистка игрового поля |  | Массив поля, [возможно что-то еще] |
| … |  |  |

### Функции модуля game

1. Частично изменим объявления и реализации функций модуля **game.** А именно, заменим одномерный массив поля на двумерный.
2. **Пока** не будем добавлять новых параметров. В качестве Y-координаты будем использовать 1, т.к. 0-вая строка используется для хранения символов верхней границы. Змейка будет двигаться только по первой строке (с индексом **1)**  двумерного массива.

### Функции модуля snake

1. Изменять объявления и реализации функций модуля **пока** не будем**.**
2. Проверьте, что программа работает корректно. (Пока змейка двигается по-прежнему горизонтально)

## Второй шаг: окончательная корректировка функций

**Задание 4**

### Функции модуля game

1. Внесите изменения в имеющиеся функции: добавьте, где требуется, массив координат змейки, направление движения змейки.

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Параметры |
| Инициализация параметров игры |  | таймаут, признак игры, [возможно что-то еще] |
| Установка змейки на поле игры |  | Массив поля игры, массив координат змейки по горизонтали,  массив координат змейки по вертикали, размер змейки,  [возможно что-то еще] |
| Проверка, что змейка съела еду |  | массив координат змейки по горизонтали ,  массив координат змейки по вертикали, параметры еды,  [возможно что-то еще] |
| Установка еды на поле игры |  | Массив поля игры, параметры еды, [возможно что-то еще] |
| Проверка конца игры |  | массив координат змейки по горизонтали,  массив координат змейки по вертикали, размер змейки,  [возможно что-то еще] |
| [ Проверка столкновения змейки с границей ] |  | Массив поля игры, массив координат змейки по горизонтали,  массив координат змейки по вертикали, [возможно что-то еще] |
| Очистка позиций змейки в игровом поле |  | Массив поля игры, массив координат змейки по горизонтали,  массив координат змейки по вертикали, [возможно что-то еще] |
| Обработка пользовательского ввода | ??? handle\_cmd (???) | Направление движения, признак окончания игры [возможно что-то еще |
| … |  |  |

### Функции модуля snake

1. Измените объявления и реализации функций**.**
2. Проверьте, что после внесения изменений программа работает корректно. (**Пока** змейка двигается по-прежнему **горизонтально**)

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Параметры |
| Инициализация параметров змейки |  | массив координат змейки по горизонтали,  массив координат змейки по вертикали, размер змейки,  [возможно что-то еще] |
| Передвижение змейки |  | массив координат змейки по горизонтали,  массив координат змейки по вертикали, размер змейки,  [возможно что-то еще] |
| … |  |  |

## Третий шаг: обработка клавиатурного ввода и управление движением змейки

### Движение змейки

Для управления змейкой используются клавиши клавиатуры. Как обрабатывать нажатие клавиш описано в самой первой работе.

**Важно:**

* При изменении направления движения важно учитывать, что змейка может поворачивать только на 90 градусов, она не может изменить направление на противоположное, т.е. не может повернуться на 180 градусов.
* Если голова змейки упирается в стенку, то движение змейки идет по кольцу, т.е. она появляется с другого (противоположного) края поля.

**Задание 5**

1. Добавьте перечисление, которое будет содержать направление движения змейки и создайте переменную этого типа в модуле main.
2. В модуле **game** объявите и определите функцию, которая будет обрабатывать нажатие клавиш. В качестве управляющих клавиш Вы можете выбрать как клавиши ‘w’ ’a’ ‘s’ ‘d’, так и клавиши стрелочек.
3. Дополнительно выполните обработку нажатия клавиши ESC. При нажатии на ESC программа должна завершаться.
4. Внесите изменения в имеющиеся функции: добавьте, где требуется, массив координат змейки, направление движения змейки.
5. Проверьте, что программа работает корректно.

### Алгоритм функции handle\_cmd [модуль game]

* Если нажата клавиша **ESC**, то выставляется флаг завершения игры.
* Если нажата клавиша, задающая направление движения змейки, то устанавливается соответствующий признак направления движения (DIRECTION)

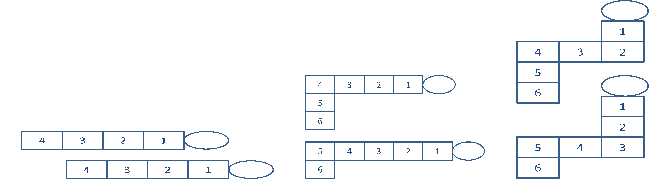
### Алгоритм функции move\_snake

* Передвигаются элементы хвоста
* Передвигается голова

Если рассматривать алгоритм укрупненно, то по сути он не изменяется. Но реально код придется дорабатывать, т.к. появлается дополнительно массив координат змейки по вертикали и признак направления движения.

Но идея первоначального алгоритма остается прежней:

* Сначала сдвигаются элементы «хвоста» . Причем, сдвиг осуществляется на одну позицию «влево». При этом происходит последовательно изменение координат частей хвоста змейки ( координаты змейки, начиная с конца, заменят "предыдущие").
* А потом в зависимости от направления движения передвигается голова, т.е. устанавливаются новые значения для координат головы по горизонтали и по вертикали.

Примеры выполнения этой операции для разных случаев приведены на рисунках 

### Алгоритм функции check\_game [модуль game]

В предыдущих проектах змейка двигалась в одномерном поле и мы проверяли игру на завершение, когда на поле негде было поставить еду (змейка достигала своего максимально возможного размера).

В этой версии проекта можно, конечно оставить это условие, но гораздо интереснее будет проверить, не укусила ли змейка саму себя.

Если координаты головы змейки равны координатам какой-либо части тела змейки, то змейка съела саму себя и надо заканчивать игру. Функция должна возвращать признак надо ли заканчивать игру.

## Совет для упрощения отладки приложения

В предыдущих проектах при анимации движения змейки применялась временная задержка между шагами. При переходе к двумерному полю отладка усложняется, поэтому рекомендуется на время отладки «автоматический» режим заменить пошаговым.

Пошаговое управление змейкой очень удобно для отладки алгоритмов передвижения змейки. После того, как будет отлажен алгоритм выполнения одного шага, можно запускать автоматическое выполнение программы.

Алгоритм одного шага игры можно представить так:

1. Генерация и установка еды
2. Очистка позиций змейки на экране
3. Проверка не съела ли еду змейка
4. Изменение положения змейки (продвижение в нужном направлении).
5. Проверка столкновения с границей поля
6. Установка новых позиций змейки в поле игры
7. Вывод поля игры на экран.
8. Проверка на конец игры
9. **Задержка выполнения игры**

Если передвижение выполняется в пошаговом режиме, то надо приостановить выполнение программы. Это можно сделать с помощью функции \_getch(). Программа будет приостановлена, пока пользователь не нажмет какую-либо клавишу. В нашей функции, осуществляющей обработку клавиатурного ввода как раз вызывается функция \_getch(). Поэтому при использовании пошагового режима работы достаточно вызвать функцию **handle\_cmd().**

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. 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** возвращает ненулевое значение, если была нажата клавиша. В противном случае возвращается значение 0.

Функция \_**kbhit** проверяет консоль на предмет недавнего нажатия клавиши. Если функция возвращает ненулевое значение, нажатие клавиши ожидает в буфере. Программа может затем вызвать \_getch для получения кода клавиши.

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 (вызов функции **handle\_cmd ()**)

**Подсказка:**

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

**Задание 6**

Проверьте, что программа работает корректно:

* змейка умеет передвигаться в разных направлениях
* змейка умеет корректно преодолевать препятствия
* размер змейки увеличивается при столкновении с едой
* при нажатии клавиши ESC происходит завершение работы программы
* завершение работы программы происходит при ситуации, когда змейка кусает саму себя.