Игра «Змейка». Отображение движения змейки в одном направлении + еда + функции с параметрами.

Темы: Инструкции цикла (while, for), перечисления, директивы препроцессора, одномерные массивы, функции с параметрами.

В этом задании мы также будем изображать движущуюся змейку, которая может съедать «еду». Но в этом проекте код необходимо распределить по функциям, причем информация в функции будет передаваться через параметры. Глобальные данные использоваться **НЕ** будут.

Использование глобальных данных не является хорошим решение, т. к. случайное изменение глобальной переменной может испортить работу всей программы. Поэтому переходим к реализации улучшенного варианта программы.

**Замечание**

Текущая версия программы все еще будет упрощенной.

* Игровое поле будет состоять только из одной строки.
* Будет производиться ограниченная обработка нажатия клавиш (для завершения игры)
* Завершение игры происходит тогда, когда на поле негде будет поставить еду (змейка достигла своего максимально возможного размера) или в том случае, если нажата клавиша ESC.
* Рабочий код будет распределен по функциям с параметрами,
* в функции main будет осуществляться **управление вызовом** этих функций. Информация в функции будет передаваться через параметры.

Текущий проект будет разрабатываться на основе предыдущего.

**Задание1**

1. Добавьте в проект файлы, те же, что и в предыдущем задании №5:

* 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".

## Данные программы

Основное отличие этой версии программы от предыдущей состоит в том, что мы **НЕ** будем использовать глобальные данные. Для передачи данных из одной функции в другую будут использоваться параметры.

Все переменные, которые нам потребуются в программе, будем определять в функции main() и передавать посредством параметров в вызываемые функции.

Константы, которые требуются для работы будут по-прежнему находиться в модуле const.

## Данные модуля const.h

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |  |
| Символ еды | food\_symbol | const.h |  |

Так как глобальные данные мы использовать не будем, то таблицы глобальных данных в этой версии проекта не потребуются.

Переработку программы начнем с изменения **сигнатур** функций.

**Замечание:**  алгоритмы функций изменяться не будут!

**Задание 2**

Измените объявления функций, добавив параметры. То, какие параметры должна иметь каждая из функций, можно определить по тому, какие данные она использует. Глобальные данные, которые использовались в предыдущем проекте, здесь преобразуем в параметры функций.

Можно использовать такую нехитрую последовательность действий:

* изучаем код функции N (в .cpp файле)
* В параметры помещаем данные, которые требуются функции (бывшие глобальные),
* Копируем заголовок функции N в соответствующий заголовочный файл.
* Если в теле функции вызывается другая функция, то корректировку начинаем с нее.

Постарайтесь заполнить таблицы для каждого модуля.

## Функции модуля snake

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

## Функции модуля field

|  |  |  |
| --- | --- | --- |
| 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 food

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Параметры |
| Инициализация параметров еды |  | Параметры еды |
| Генерация еды |  | Позиция еды на поле |
| … |  |  |

### Алгоритм функции init\_food( )

Пока эта функция является «заглушкой».

Так как параметры еды задаются в модуле main, в этой версии проекта будем выводить на консоль полученные значения параметров.

Кроме того, можно проверить эти значения на корректность.

### Алгоритм функции generate\_food( )

Генерируется случайное значение, которое используется в качестве координаты еды в поле игры. Важно, чтобы позиция еды попадала в индексов игрового поля, но в то же время не попадала на границу поля.

## Функции модуля game

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Параметры |
| Инициализация параметров игры |  | таймаут, признак игры, [возможно что-то еще] |
| Установка змейки на поле игры |  | Массив поля игры, массив змейки , размер змейки,  [возможно что-то еще] |
| Проверка, что змейка съела еду |  | массив змейки , параметры еды,  [возможно что-то еще] |
| Установка еды на поле игры |  | Массив поля игры, параметры еды, [возможно что-то еще] |
| Проверка конца игры |  | массив змейки, размер змейки,  [возможно что-то еще] |
| Проверка столкновения змейки с границей |  | Массив поля игры, массив змейки, [возможно что-то еще] |
| Очистка позиций змейки в игровом поле |  | Массив поля игры, массив змейки, [возможно что-то еще] |
| ... |  |  |

### Алгоритм функции init\_game( )

Пока эта функция является «заглушкой».

Так как параметры игры задаются в модуле main, в этой версии проекта будем выводить на консоль полученные значения параметров.

Кроме того, можно проверить эти значения на корректность.

Алгоритмы остальных функций модуля game не изменяются.

## Функция main

Все данные (кроме констант) наши функции будут получать теперь из функции main(). Их мы будем формировать локально.

**Задание 3**

1. Определите переменные, которые используются в функции main ().

Это сделать не очень сложно. Надо просто создать локальные переменные, аналогичные глобальным.

Можно, конечно, просто перенести все глобальные переменные в функцию main (), но для создания условий, приближенных к реальным, лучше создать переменные, которые будут иначе (но похоже) именоваться.

Например, **snake\_size**  и **snake\_size\_m**. Это поможет избежать ошибок при передаче аргументов в функции. (суффикс **\_m** в примере означает main)

1. Заполните следующую таблицу.

## Данные (переменные) функции main

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

1. Удалите глобальные переменные из модулей

### Алгоритм функции main (один из возможных)

1. Определение переменных, которые используются в игре
2. Подготовка к игре
   1. инициализация игры (вызов функции **init\_game ()** с параметрами)
   2. инициализация игрового поля (вызов функции **init\_field ()** с параметрами)
   3. инициализация змейки (вызов функции **init\_snake ()** с параметрами)
   4. [ инициализация еды (вызов функции **init\_food ()** с параметрами) ]
   5. “Установка” змейки в поле игры (вызов функции **set\_snake ()** с параметрами)
   6. Вывод поля игры на экран (вызов функции **print\_field ()** с параметрами)
   7. Вывод приглашения к игре
3. В цикле, пока идет игра выполняется алгоритм одного шага
   1. Установка еды на поле (вызов функции **set\_food ()** с параметрами)
   2. Очистка старых позиций змейки в поле (вызов функции **clear\_snake ()** с параметрами)
   3. Проверка не съела ли еду змейка (вызов функции **check\_eating ()** с параметрами)
   4. Передвижение змейки (вызов функции **move\_snake ()** с параметрами)
   5. Проверка не уперлась ли змейка в граничный элемент (вызов функции **check\_snake ()** с параметрами)
   6. Установка новых позиций змейки в поле игры (вызов функции **set\_snake ()** с параметрами)
   7. Вывод поля игры на экран (вызов функции **print\_field ()** с параметрами)
   8. Проверка завершения игры (вызов функции **check\_game ()** с параметрами )
   9. Временная задержка (**Sleep** () )
4. Завершение игры

**Задание 4**

Проверьте, что после внесения всех корректировок программа работает корректно:

* змейка умеет передвигаться в горизонтальном направлении (влево, вправо)
* змейка умеет корректно преодолевать препятствия
* при поедании еды размер змейки увеличивается на 1
* Если змейка достигла своего максимального размера (на поле нет места для установки еды), происходит завершение работы программы
* при нажатии клавиши ESC происходит завершение работы программы