Игра «Змейка». Движение змейки в одном направлении + еда + функции без параметров.

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

В этом задании мы также будем изображать движущуюся змейку, которая может съедать «еду», но в код в этом проекте необходимо распределить по функциям.

**Замечание**

Текущая версия программы все еще будет упрощенной.

* Игровое поле будет состоять только из одной строки.
* Обработка нажатия клавиш производиться не будет
* Завершение игры происходит тогда, когда на поле негде будет поставить еду (змейка достигла своего максимально возможного размера).
* рабочий код будет распределен по функциям, а в main будет осуществляться только управление вызовом этих функций.
* Однако, пока все функции будут без параметров, т.е. сохраняется принцип использования глобальных переменных для обмена данными между модулями.

Текущий проект будет разрабатываться на основе предыдущего.

**Задание 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".

## Глобальные данные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание | переменная | | Модуль | | Примечания |
| Текущий размер змейки | snake\_size | | snake | | >0 |
| 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-2 (2 граничных элемента) |
| Массив координат змейки по горизонтали (X) | snake\_x[L] | | snake | | >=0 |
| Массив игрового поля | field [columns] | | field | | >=0, пока одномерный |
| Символ еды | food\_symbol | const.h | |  | |
| Признак, что еда установлена | food\_flag | food | |  | |
| Координата еды по горизонтали (X) | food\_x | food | |  | |
| Признак продолжения игры | game\_on | game | |  | |

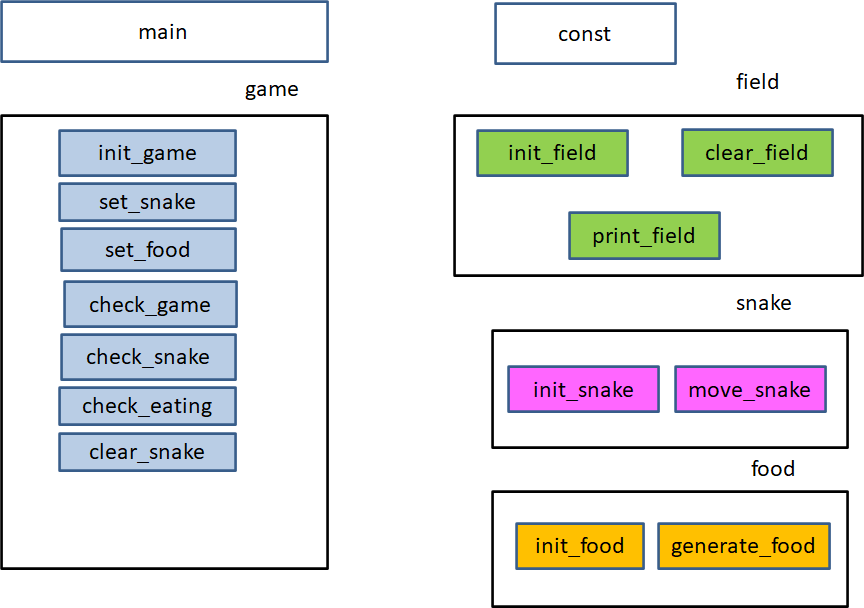
**Задание 2**

Постарайтесь выделить повторяющиеся фрагменты в коде функции main. Еще раз подумайте о декомпозиции всей задачи и постарайтесь создать функции, которые будут включать эти повторяющиеся фрагменты. Поместите функции в соответствующие модули, а в функции main заместите выбранные фрагменты вызовом соответствующих функций.

## Схема декомпозиции задачи

Вам предлагается один из вариантов декомпозиции задачи. Вам может показаться, что некоторые функции лишние (init\_..., check\_...), но это задел на будущее.

Вы можете выполнить декомпозицию иначе. При этом необходимо будет описать свои функции в соответствующих табличках.



**Замечания:**

1. Не стоит сразу весь код изменять. Лучше делать это постепенно, каждый раз проверяя то, что получилось.
2. Постарайтесь заполнить следующие таблицы

Рекомендация: Информацию о данных того или иного модуля можно взять из таблицы глобальных данных, которая приведена выше.

## Модули программы

|  |  |
| --- | --- |
| Название | Описание |
| snake |  |
| game |  |
| common |  |
| field |  |
| food |  |
| main |  |

## Данные модуля const.h

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип | Примечания |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Данные модуля main.cpp

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип | Примечания |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Данные модуля snake

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип | Примечания |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Данные модуля field

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип | Примечания |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Данные модуля game

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип | Примечания |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Данные модуля food

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Идентификатор | тип | Примечания |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Функции модуля snake

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Примечания |
| Инициализация параметров змейки |  |  |
| Передвижение змейки |  |  |
| … |  |  |

### Алгоритм функции init\_snake ( )

* Устанавливается начальный размер змейки.
* Массив змейки заполняется значениями

### Алгоритм функции move\_snake ( )

* Передвигаются элементы хвоста
* Передвигается голова

## Функции модуля field

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Примечания |
| Вывод поля на экран |  |  |
| Инициализация поля |  |  |
| Очистка игрового поля |  |  |
| … |  |  |

### Алгоритм функции init\_field( )

* Граница поля заполняется символом границы (например, #).
* Вызов функции **clear\_field()**;

### Алгоритм функции print\_field( )

* Очистка экрана
* Массив игрового поля выводится на экран.

Замечание :

При реализации прежних версий проекта для очистки экрана мы использовали вызов функции **system("cls");**

При этом экран очищался, но периодическое «мигание» консоли не добавляло положительных эмоций.

В текущей (и следующих) версиях Вам предлагается вместо вызова **system("cls")**; для очистки экрана при перерисовке поля игры использовать функцию **setCursorPosition()**.

Файл реализации и файл интерфейса прилагается к заданию.

### Алгоритм функции clear\_field( )

Массив игрового поля заполняется символом-заполнителем (например, пробелом).

## Функции модуля food

|  |  |  |
| --- | --- | --- |
| Описание | Идентификатор | Примечания |
| Инициализация параметров еды |  |  |
| Генерация еды |  | Функция не является необходимой |
| … |  |  |

### 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.

### Алгоритм функции init\_food()

Устанавливаются начальные значения для параметров еды.

## Функции модуля game

|  |  |  |
| --- | --- | --- |
| Описание | Описание (сигнатура) функции | Примечания |
| Инициализация параметров игры |  |  |
| Установка змейки на поле игры |  |  |
| Проверка, что змейка съела еду |  |  |
| Установка еды на поле игры |  |  |
| Проверка конца игры |  | Проверяем, что змейка достигла своего максимального размера |
| Проверка столкновения змейки с границей |  |  |
| Очистка позиций змейки в игровом поле |  |  |
| ... |  |  |

### Алгоритм функции init\_game( )

Устанавливаются начальные значения параметров игры.

### Алгоритм функции set\_snake( )

Змейка устанавливается в игровом поле, т.е.

* символ головы змейки копируется в соответствующую позицию игрового поля
* символы хвоста также копируются в игровое поле.

### Алгоритм функции clear\_snake( )

Очищается позиция змейки в массиве, т.е. в соответствующей позиции поля устанавливается символ-заполнитель поля.

### Алгоритм функции set\_food ( )

Еда устанавливается в игровом поле. Можно вызывать функцию generate\_food() и проверять, что в сгенерированной позиции ничего нет.

### Алгоритм функции check\_eating ( )

Выполняется проверка того, что змейка съела еду. Если координаты еды и координаты головы змейки совпадают, то устанавливается в false признак того, что еда есть на поле.

### Алгоритм функции check\_snake ( )

Выполняется проверка того, что змейка встретилась с границей поля.

Важно!

* Если голова змейки упирается в стенку, то движение змейки идет по кольцу, т.е. она появляется с другого (противоположного) края поля.

### Алгоритм функции check\_game ( )

Проверка завершения игры.

Завершение игры происходит тогда, когда на поле негде будет поставить еду (змейка достигла своего максимально возможного размера).

### Алгоритм функции main (один из возможных)

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