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

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

В этом задании мы также будем изображать движущуюся змейку, но в дополнение к змейке на поле появляется еда.

1. Еда для змейки возникает на поле в случайном (но только **пустом**) месте.
2. Если змейке удается «съесть» еду (координаты головы змейки совпадают с координатами еды), то длина змейки увеличивается на 1 элемент.

**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)
* Весь рабочий код пока будет располагаться в функции main.
* Завершение игры происходит тогда, когда на поле негде будет поставить еду или если нажата клавиша ESC.

В текущей работе, как и в работе «Отображение змейки\_3» будем применять одномерные массивы.

Текущий проект будет разрабатываться на основе предыдущего.

**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 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".

1. Добавьте в проект новые файлы:

food.cpp и food.h, которые будут содержать информацию, относящуюся к понятию "Еда".

1. В файл const.h добавьте константу для отображения символа еды

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

## Глобальные данные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Описание | переменная | | Модуль | | Примечания |
| Текущий размер змейки | snake\_size | | snake | | >0 |
| Таймаут задержки между шагами игры | timeout | | game | | >100 (ms) |
| Массив координат по горизонтали (X) хвоста змейки | snake\_x[L] | | snake | | >=0 |
| Массив игрового поля | field [columns] | | field | | >=0, пока одномерный |
| Признак продолжения игры | game\_on | game | |  | |
| Направление движения змейки | direction | snake | | В качестве типа для переменной  удобно использовать перечисление, например, DIRECTION | |
| Признак, что еда установлена | food\_flag | food | |  | |
| Координата еды в поле игры (по горизонтали) | food\_x | food | |  | |

**Задание 2**

Определите и проинициализируйте в соответствующих модулях нужные переменные и константы:

* Символ еды
* Признак, что еда установлена
* Координаты еды в поле игры

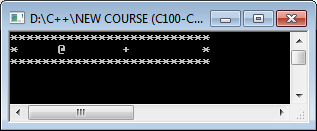
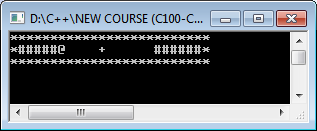
**Генерация еды**

* Позиция еды определяется с помощью генерации случайного значения. Причем это значение должно попадать в пределы поля и в то же время эта позиция поля не должна быть занята (ни граничными символами, ни символами змейки). Позиция еды не меняется до тех пор, пока змейка не съест ее.
* В том случае, если змейка “съела” еду, требуется сгенерировать новую позицию еды.
* Выполнять генерацию имеет смысл до тех пор, пока есть место на поле, куда можно поставить еду. Если такого места нет (а это возможно, когда змейка достигла своего максимально возможного размера), то нужно заканчивать игру.

Упрощение: если на текущем шаге игры не удалось добавить еду на поле игры, то можно будет это сделать на следующем шаге.

**Задание 3**

* Сгенерируйте еду и установите ее на поле.
* В том случае, если змейке удается «съесть» еду (координаты головы змейки совпадают с координатами еды), то длину змейки надо увеличить на 1 элемент. При этом надо будет пересчитать элементы «хвоста» и передвинуть змейку.
* В том случае, если змейке удается «съесть» еду, необходимо установить в false флаг наличия еды в поле игры
* При условии, что еда съедена, необходимо сгенерировать значение для новой позиции еды и установить еду на поле игры.

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. Если автоматический режим работы, то
   * Временная задержка (**Sleep()**)

* Проверка, нажата ли какая-либо клавиша (вызов **\_kbhit()),**

если нажата клавиша ESC, то сбрасываем флаг продолжения игры,

иначе, ничего не делаем (пока)

1. Если пошаговый режим работы, то

если нажата клавиша ESC, то сбрасываем флаг продолжения игры,

иначе, ничего не делаем (пока)

### Движение змейки

Также, как и в предыдущем проекте, движение змейки может выполняться в разных режимах:

* Пошаговом
* Автоматическом

Пошаговое управление змейкой очень удобно для отладки алгоритмов передвижения змейки. Так как в игру добавляется новая функциональность, то начать отладку следует с пошагового режима. После того, как будет отлажен алгоритм выполнения одного шага, можно запускать автоматическое выполнение программы.

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

### Укрупненно алгоритм программы может выглядеть следующим образом

Diagram

Description automatically generated

**Задание 6**

Проверьте, что 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