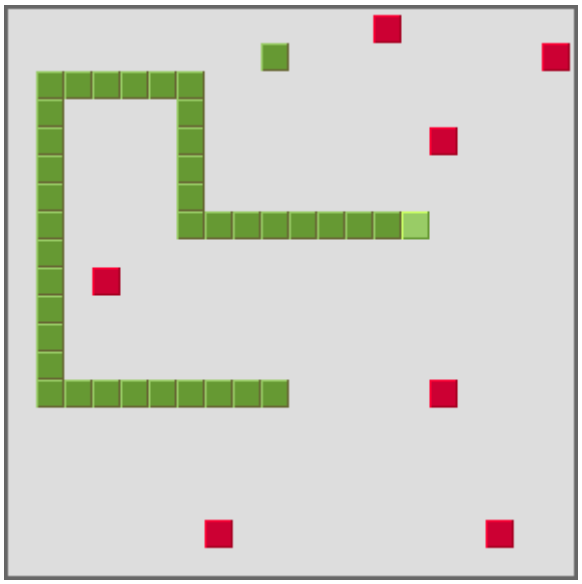
ИГРА

ПИТОН

.snake (Питон, Удав, Змейка и др.) — компьютерная игра, возникшая в середине или в конце 1970-х.

Игрок управляет змейкой, которая ползает по плоскости (как правило, ограниченной стенками), собирая еду (или другие предметы), избегая столкновения с собственным хвостом и краями игрового поля. Каждый раз, когда змея съедает кусок пищи, она становится длиннее, что постепенно усложняет игру. Игрок управляет направлением движения головы змеи (обычно 4 направления: вверх, вниз, влево, вправо), а хвост змеи движется следом. Игрок не может остановить движение змеи.



Указания:

Реализуйте игру для текстового терминала. Пусть каждый уровень будет размером 40 на 20 клеток. В каждой клетке может находиться либо стена (#), либо еда(@),либо клетка свободна (.). Питона изобразите в виде последовательности буков “O”. Раскраска уровня по вашему усмотению.

Уровень на рисунке 1 в нашем случае будет выглядеть так:

```
########################################
#.............############..........#
#.............############..........#
#.............############..........#
#.............############..........#
#......................................#
#......................................#
#......................................#
#......................................#
#............@.........................#
#......................................#
#........................#########.....#
#........................#########.....#
#........................#########.....#
#........................#########.....#
############............................#
############............................#
############............................#
############............................#
########################################
```

Рис. 1. Первый уровень игры. Вид файла уровня.

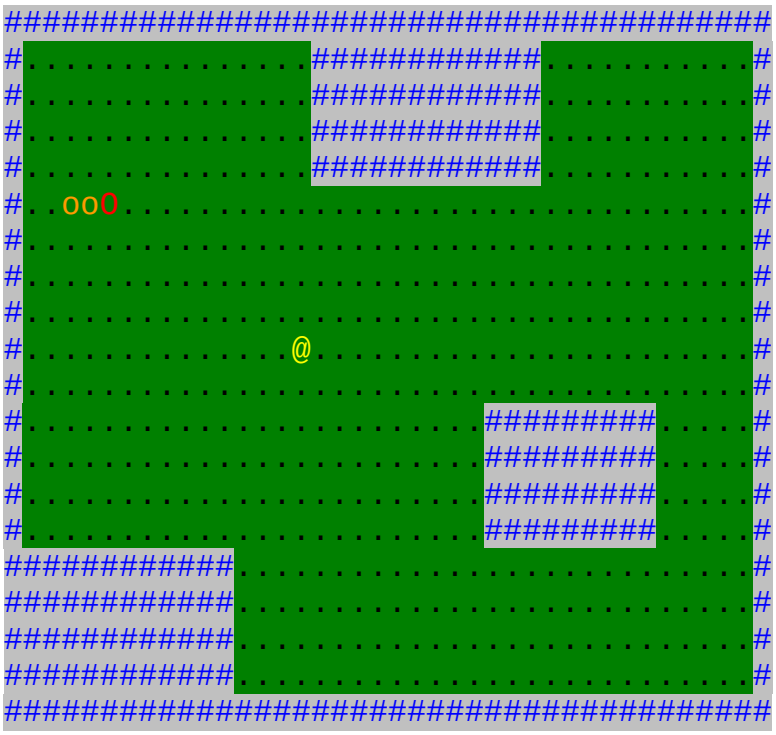


Рис. 2. Первый уровень игры. Вид на экране.

В начальный момент питон появляется в клетке с координатами (5;5), его длина 3 клетки, а хвост расположен слева.

Чтобы написать эту игру вам необходимо знать принципы работы с переменными, массивами, файлами, вывод цветного текста на экран и уметь работать с функциями.

Создайте файл уровня. Это должен быть простой текстовый файл, расположенный в одной директории с исполняемым файлом вашей программы. Назовите его **level1.txt**. Содержимое файла представлено на рисунке 1.

Объявите в глобальном пространстве

```
char Field[40][20];        // Здесь будем хранить сам игровой уровень
int snake[100][2];         // Здесь будем хранить координаты тела питона
char TempField[40][20];    // Этот массив понадобится для обрисовки игрового уровня и питона
int SnakeDlina=3;          // Эта переменная будет хранить длину нашего питона в клетках
```

Сделайте процедуру чтения содержимого файла **level1.txt** в массив **Field[40][20]**. Назовём её **LoadLevel()**.

Реализуйте процедуру вывода уровня на экран **PrintField()**.

На нашем уровне нет питона. Будем хранить его координаты в массиве snake. В **snake[0]** координаты головы, в snake[1] координаты следующей ячейки питона и т.д. при этом в **snake[][0]** будем помещать координату x, в **snake[][1]** будем помещать координату y. Заполните в процедуре **void LoadLevel()** массив snake согласно начальному условию.

Чтобы отрисовать игровой уровень и питона на нём, модифицируйте процедуру **void PrintField()**. Скопируйте содержимое Field[40][20] в TempField[40][20] и поместите изображение питона в **TempField[40][20]**. Затем отрисуйте **TempField[40][20]**.

Чтобы питон двигался, надо поместить отрисовку уровня в цикл. Каждый раз перед рисованием уровня следует очищать экран командой **system("clear")**. После рисования уровня следует считать символ введённый пользователем с клавиатуры. Условием выхода из этого цикла сделайте то, что пользователь ввёл символ “q”.

Далее голова удава должна переместиться на одну клетку вперёд (согласно выбранному пользователем направления). Соответственно все клетки удава должны сдвинуться в направлении головы. Для этого реализуйте процедуру **void MoveSnake(int ch)**, где **ch** это введённый пользователем символ. Объявите в программе переменную, в которой будете хранить введённый пользователем символ, и сразу присвойте ей значение “d” (перемещаться влево).

В процедуре **MoveSnake** в цикле местите хранимые в **i-1**ой ячейке массива **snake** в **i**ую для **i** от **SnakeDlina** до **1**. А в ячейку **snake[0]** запишите теже координаты, что в ней хранятся (это положение головы питона на предыдущем шаге), смещённый вверх, вниз, вправо или влево, в зависимости от того, какой символ ввёл пользователь (удобно использовать **W,A,S** и **D**, как указания сдвинуться вверх, влево, вправо и вниз соответственно.

Сделайте процедуру проверки того, что голова питона оказалась в клетке с едой **void CheckFood()**. Если это так, то увеличте длину питона на 1 и переместите еду в место с произвольными координатами. При выборе координат нового положения еды, следи за тем, чтобы это было пустое, не занятое стенами место на карте уровня. Для этого можно сделать цикл, в котором новые координаты будут генерироваться до тех пор, покаони не удовлетворят нашему условию. На прежнее место нахождения еды следует поместить точку (.).

Реализуйте процедуру проверки, не выполнил ли пользователь действия, приводящие к завершению игры **bool CheckGameOver()**, а именно – не врезался ли питон в стену и не укусил ли сам себя за хвост. Эта процедура должна вернуть истину, если питон наполз на стену (координаты головы совпали с координатами стены) или укусл себя за хвост (координаты головы совпали с одной из координат его тела).

Добавьте **CheckGameOver()** к условию выхода из основного игрового цикла. Также добавьте к условию выхода ситуацию достижения питоном длины большей 15 (будем считать это победой).

После игрового цикла в зависимости от того равна ли длина питона 15 или нет реализуйте вывод на экран надписи “вы выиграли” либо “вы проиграли”.

Теперь замените оператор чтения символа с клавиатуры на оператор задержки на одну секунду (sleep(1)) и последующую проверку в цикле нажата ли клавиша и если да, то чтение её кода до тех пор, пока клавиша окажется не нажатой. Цикл нужен на тот случай если нетерпеливый пользователь нажмёт в течении секунды ожидания несколько клавиш подряд. **while (kbhit()) ch= readch();**

Функции **kbhit()** и **readch()** возьмите из библиотеки kbhit.h (скопируйте **http://rff.tsu.ru/~nick/FizMat11/misc/kbhit.h** и **http://rff.tsu.ru/~nick/FizMat11/misc/kbhit.cpp** в папку проекта и в самом начале файла программы добавьте **#include "kbhit.h"**). В начале программы добавьте **init_keyboard();** в конце **close_keyboard();**

Поздравляю! Игра готова!