Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Сибирский государственный университет телекоммуникаций и информатики»

(ФГОБУ ВПО «СибГУТИ»)

КУРСОВАЯ РАБОТА

по дисциплине “Программирование” на тему

“Змейка”

Выполнил:

студент гр. ИП-711

Мартасов Илья Олегович

Проверил:

Доцент

кафедры ПМ и К

Перцев Игорь

Владимирович

Новосибирск

2018

Содержание

1.Постановка задачи и описание игры…..………………………………………3

2.Описание алгоритма…………………………………………………………….4

3.Результат работы программы…………………………………………………..5

4.Код программы………………………………………………………………….8

**1.Постановка задачи и описание игры.**

Задача, которую предстояло решить – реализация игры “Змейка”.

По экрану передвигается цепочка символов, постоянно удлиняющаяся при наезде на так называемый “фрукт”, который появляется в различных местах экрана, и при этом не выходящий за границы отведенного для игры поля. Цель игры – собрать как можно больше “фруктов” и при этом не столкнуться с препятствиями в виде стен и собственного тела. За каждый собранный “фрукт” начисляется 10 очков. Чем больше набрано очков, тем лучше. Игра идет до тех пор, пока змейка не наедет на саму себя, или же пока она не врежется в одну из четырех стен, которые ограничивают поле игры. Также возможен случай, когда змейка полностью заполнит собой всё игровое поле, но это происходит крайне редко.

В игре предусмотрены три уровня сложности: простой, средний и сложный. От его выбора зависит скорость передвижения змейки – чем сложнее, тем быстрее. По умолчанию установлен простой уровень.

Игра предназначена для спокойного времяпровождения, а наличие подсчёта очков предоставляет возможность пользователям устраивать небольшие соревнования между собой.

Программа реализована на языке C++ полностью в консольном виде.

**2.Описание алгоритма**

Поскольку вся игра проходит в консоли программы, то для корректного использования консольного окна берется его дескриптор:

HANDLE console = GetStdHandle (STD\_OUTPUT\_HANDLE);

с помощью которого возможен вывод информации на экран, перемещение курсора по всей консоли (SetConsoleCursorPosition), а также установка атрибутов (в данном случае, цвета) символов(SetConsoleTextAttribute).Для перемещения курсора по консоли также используется функция gotoxy. В неё передаются координаты x и y. Они записываются в переменную place типа COORD (short X; short Y;) , которая затем используется в фунцкии перемещения курсора вместе с дескриптором.

Тело змейки состоит из структуры, состоящей из двух полей: COORD \*t, в которой хранятся координаты каждой точки змейки и int score – количество съеденных змейкой “фруктов”. Также используется структура, хранящая данные о самой змейке – её тело, координаты расположения “фрукта”, направление движения змейки по осям x и y и указатель на направление, в котором змейка движется. Все поля передаются через переменную value. Для обозначения направления движения, а также конца игры, съедания “фрукта” и движения змейки используется enum values.

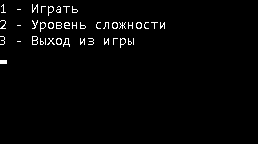
Когда начинается игра, первым делом очищается экран и устанавливаются начальные положения змейки и “фрукта”(функция start) – исходные координаты змейки, её длина(по умолчанию 4), и направление движения(по умолчанию – вправо).Для “фрукта” (функция addfruit) координаты вычисляются случайным образом, но при этом происходит проверка на то, чтобы “фрукт” не появился на самой змейке. Когда проверка пройдена, курсор переносится в сгенерировавшиеся координаты и на их месте рисует “фрукт”.

После этого рисуется поле игры – при помощи gotoxy курсор переносится в нужную точку консоли, и затем рисуются горизонтальные и вертикальные полосы, ограничиваемые константами weight и height (60 и 25 соответственно). Функция logic отвечает за логику игры – в ней происходят: проверка на столкновение змейки с препятствиями(return end), проверка на сбор “фрукта”(return point), а также визуализация передвижения змейки по экрану(return move). Само движение реализовано в функции moving. Оно происходит при помощи стрелок на клавиатуре, а для того, чтобы змейка не разворачивалась вокруг своей оси, проверяется, в каком направлении движется змейка и можно ли ей двигаться вообще.

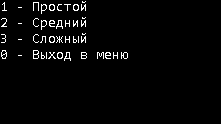
Скорость движения змейки определяется при помощи функции Sleep, куда передаётся значение задержки, при которой змейка движется.

**3.Результат работы программы**

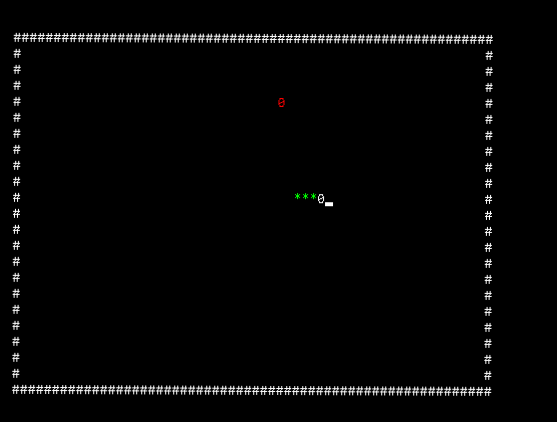
При запуске игры, на экране появляется консоль, на которой высвечено небольшое меню:



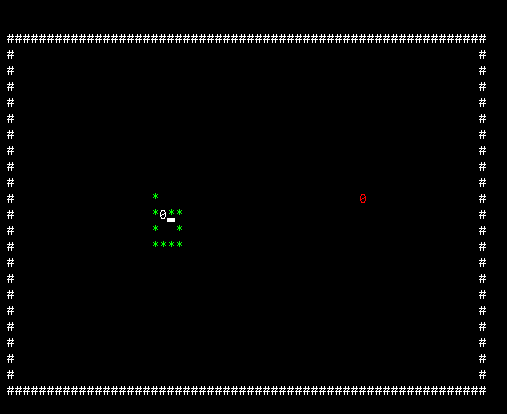
Выбрав пункт “Уровень сложности”, пользователь увидит три возможных варианта сложности игры, а также вариант выхода в меню:



Когда пользователь запустит игру, на экране высветится поле со змейкой и фруктом, расположенным в случайном месте:



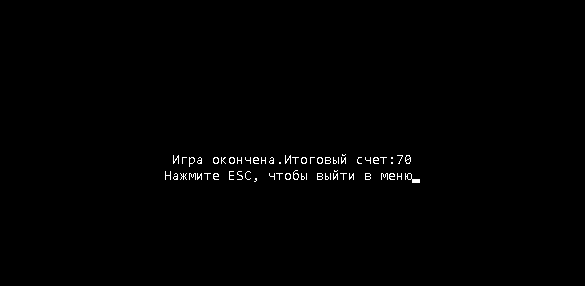
Если во время игры змейка столкнется сама с собой:



Или врежется в стену:



На экране высветится надпись, сообщающая о конце игры и выводящая итоговый счет пользователя:



После выхода в меню, пользователь сам решает, что ему делать дальше: начать новую игру, изменить сложность или же выйти.

**4. Код программы**

#include <iostream>

#include <ctime>

#include <windows.h>

#include <conio.h>

using namespace std;

HANDLE console = GetStdHandle(STD\_OUTPUT\_HANDLE);

const int weight = 60;

const int height = 25;

enum values {UP, DOWN, LEFT, RIGHT, STOP, end, point, move};

struct body{

COORD \*t;

int score;

};

struct snake{

body piton;

COORD fruit;

int dx;

int dy;

int move;

}value;

void gotoxy(int x, int y)

{

COORD place = {x, y};

SetConsoleCursorPosition(console, place);

}

void field()

{

gotoxy(2, 2);

for(int i = 0; i < weight; i++)

cout<<"#";

gotoxy(2, height - 1);

for(int i = 0; i < weight; i++)

cout<<"#";

for(int i = 3; i < height; i++)

{

gotoxy(2, i);

cout <<"#"<<endl;

}

for(int i = 3; i < height; i++)

{

gotoxy(weight + 1, i);

cout <<"#"<<endl;

}

}

void addfruit(snake &value)

{

int i;

int n = value.piton.score;

do{

value.fruit.X = rand() % 56+3;

value.fruit.Y = rand() % 19+3;

for(i = 0; i < n; i++)

if(value.fruit.X == value.piton.t[i].X && value.fruit.Y == value.piton.t[i].Y)

break;

}while(i < n);

SetConsoleCursorPosition(console, value.fruit);

SetConsoleTextAttribute(console, 0x0c);

cout<<"0";

SetConsoleTextAttribute(console, 0x0f);

}

void start(snake &value)

{

value.piton.score = 4;

value.piton.t = new COORD [4];

for(int i = 0; i < 4; i++)

{

value.piton.t[i].X = weight / 2 + i;

value.piton.t[i].Y = height / 2;

}

value.move = RIGHT;

value.dx = 1;

value.dy = 0;

addfruit(value);

}

int logic(snake &value)

{

int &n = value.piton.score;

COORD head = value.piton.t[n - 1];

COORD tail = value.piton.t[0];

COORD next;

next.X = head.X + value.dx;

next.Y = head.Y + value.dy;

for(int i = 0; i < n; i++)

if((next.X == value.piton.t[i].X && next.Y == value.piton.t[i].Y )

|| (next.X < 3 || next.Y < 3 || next.X > weight || next.Y > height - 2))

{

value.move = STOP;

return end;

}

if(next.X == value.fruit.X && next.Y == value.fruit.Y)

{

COORD \*temp = new COORD[ ++n ];

for(int i = 0; i < n; i++)

temp[i] = value.piton.t[i];

temp[n - 1] = next;

delete [] value.piton.t;

value.piton.t = temp;

SetConsoleCursorPosition(console, head);

SetConsoleTextAttribute(console, 0x0a);

cout<<"\*";

SetConsoleCursorPosition(console,next);

cout<<"0";

addfruit(value);

return point;

}

for(int i = 0; i < n - 1; i++)

value.piton.t[i] = value.piton.t[i + 1];

value.piton.t[n - 1] = next;

SetConsoleCursorPosition(console, tail);

cout<<" ";

SetConsoleCursorPosition(console, head);

SetConsoleTextAttribute(console, 0x0a);

cout<<"\*";

SetConsoleCursorPosition(console, next);

SetConsoleTextAttribute(console, 0x0f);

cout<<"0";

return move;

}

void moving()

{

int move = getch();

if(move == 72 && value.move != DOWN && value.move != STOP)

{

value.move = UP;

value.dx = 0;

value.dy = -1;

}

else if(move == 80 && value.move != UP && value.move != STOP)

{

value.move = DOWN;

value.dx = 0;

value.dy = 1;

}

else if(move == 75 && value.move != RIGHT && value.move != STOP)

{

value.move = LEFT;

value.dx = -1;

value.dy = 0;

}

else if(move == 77 && value.move != LEFT && value.move != STOP)

{

value.move = RIGHT;

value.dx = 1;

value.dy = 0;

}

}

int menu()

{

int k = 100;

m:system("cls");

SetConsoleTextAttribute(console, 0x0f);

cout<<"1 - Играть"<<endl;

cout<<"2 - Уровень сложности"<<endl;

cout<<"3 - Выход"<<endl;

int score = 0;

int t = getch();

switch(t){

case '1':

system("cls");

start(value);

field();

while(1)

{

while(!kbhit())

{

switch (logic(value))

{

case point:

score += 10;

break;

case end:

Sleep(1000);

system("cls");

gotoxy(25, 10);

cout<<"Игра окончена. Итоговый счет:"<<score<<"\n\t\t\tНажмите ESC, чтобы выйти в меню";

if(getch() == 27)

goto m;

break;

}

Sleep(k);

}

moving();

}

break;

case '2':{

system("cls");

cout<<"1 - Простой"<<endl;

cout<<"2 - Средний"<<endl;

cout<<"3 - Сложный"<<endl;

cout<<"0 – Выход в меню"<<endl;

int s = getch();

switch(s){

case '1':

k = 100;

goto m;

break;

case '2':

k = 50;

goto m;

break;

case '3':

k = 25;

goto m;

break;

case '0':

goto m;

break;

}

}

break;

case '3':return 0;

}

}

main()

{

setlocale(LC\_ALL, "Rus");

srand(time(NULL));

menu();

}