

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Пояснительная записка

к курсовой работе

по курсу «Программирование»

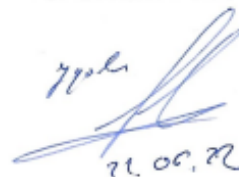
на тему «Разработка программы сложной структуры методом нисходящего
программирования»

Выполнил:

Кузнецов А.

Принял:

Голотенков Н.О.



21.06.22

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ



М.А. Митрохин

« ____ » _____ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Митрохину Алексею Сергеевичу Группа 21ВВ1

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

*Приложение: игровая программа "Змейка".
Требования: графический интерфейс, не менее 15 клеток по горизонтали и вертикали, функция ввода шипа, запись результата в файле, заставка при загрузке.*

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "22" февраля

Дата защиты проекта "22" окт. 22

Руководитель Галатенков Н.О.

Задание получил "22" февраля 2022г.

Студент Музиезов И.С.

1 Постановка задачи и обязательные требования.

Необходимо разработать программу «Игра змейка», которая должна отвечать следующим требованиям:

1. Многомодульность. Программа должна быть поделена на логические модули. Это упростит дальнейшее совершенствование программы, а также позволит локализовать ошибки.
2. Использование сложных типов данных: массивов, структур, объектов и т.д.
Для разработки данной игры необходимо подробно изучить принципы работы с массивами, структурами и объектами для создания игрового поля и реализации графики.
3. Режим работы видеосистемы: текстовый/графический. Необходимо определиться с типом управления, видом интерфейса и изучить приёмы реализации графики.
4. Устройство ввода информации: клавиатура/мышь. Необходимо определиться с механикой взаимодействия игрока с игрой и её интерфейсом.
5. Пользовательский интерфейс должен быть построен на основе меню или панели инструментов. Необходимо реализовать пользовательское меню, с заставкой.
6. Операционная система: MS Windows. При разработке использовать среду MS Windows с системой программирования MS Visual Studio.
7. Язык программирования: Си и Ассемблер.

2.Описание структуры программы

Файл “Project5” – является главным файлом . В нём объявляются основные переменные ,структуры , массивы и описываются основные функции.

Функция “main” - является главной функцией , в которой вызываются функции описывающие игровой процесс, задается считывание клавиш , запись в файл и чтение с файла.

Структура:



Основной файл “Project5” содержит в себе все основные функции , такие как

“game” – функция, которая рисует границы и заполняет внутренность поля ,
“gen_food” – функция которая генерирует яблоки рандомным образом ,
“createsnake” – эта функция отрисовывает змейку , “xvost_ypr” – функция ,
которая отвечает за движение змейки , “xvost” – функция , которая отвечает
за движения хвоста , “food” – функция , которая отвечает за собирание яблок
и увеличение рекорда .

Заголовочный файл “oneoffunc.cpp” содержит в себе функцию hidecursor,
которая нужна для очистки фона от курсора.

3. Описание функционала программы

Функция “main” -главная функция. В ней реализуется вызов всех функций ,
запись и чтения с файла , сравнение рекорда , считывание клавиш , а также
некоторая часть из логики игры.

Сначала идет считывание информации рекорда из файла, затем очистка
курсора по вызову функции “hidecursor” , далее вставка на ассемблере ,
которая определяет середину для координат x , y . Потом запускается цикл ,
который работает пока змейка не врежется в себя или в заданные границы
, затем идет очистка координат при помощи функции “clear” , создание
границ при помощи функции “game” , потом генерация фруктов “gen_food”
, далее создание самой змейки по вызову функции “createsnake” , потом
отрисовка границ по вызову функции “printgame” , затем функция
“xvost_ypr” регулирует движение змейки , функция “xvost” , регулирует
движения хвоста , по вызову функции “food” определяется результат
столкновения головы змейки с фруктом. Ниже представлена часть кода ,
которая отвечает за считывание клавиш и не дает игроку двигаться в
противоположные направления.

```
if (GetKeyState(nVirtKey: 'L') < 0)(gameover = true);  
if (qas != down) { if (GetKeyState(nVirtKey: 'W') < 0)(qas = up); }  
if ((qas != up) and (qas != 0)) { if (GetKeyState(nVirtKey: 'S') < 0)(qas = down); }  
if (qas != right) { if (GetKeyState(nVirtKey: 'A') < 0)(qas = left); }  
if (qas != left) { if (GetKeyState(nVirtKey: 'D') < 0)(qas = right); }
```

Далее происходит сравнение текущего результата с рекордом , и запись в
файл если текущий результат больше рекорда.

```

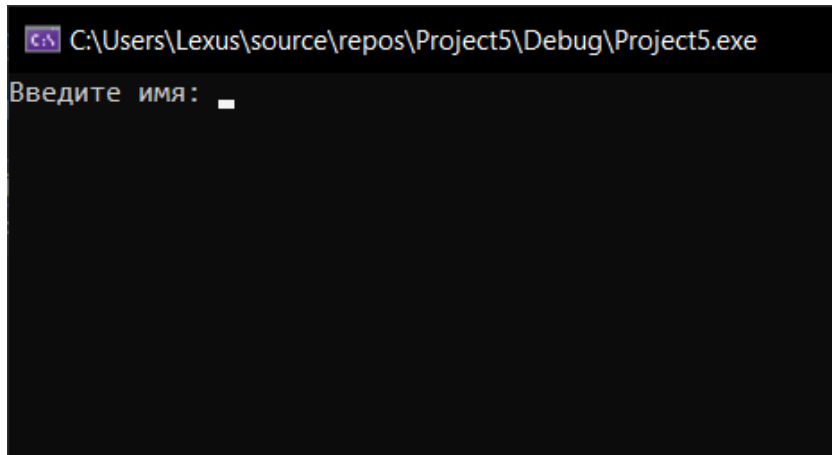
if (score > rec_m)
{
    file = fopen(_FileName: "D:/SaveSnake.txt", _Mode: "w");
    fprintf(_Stream: file, _Format: "%d", score);
    fclose(_Stream: file);
}

```

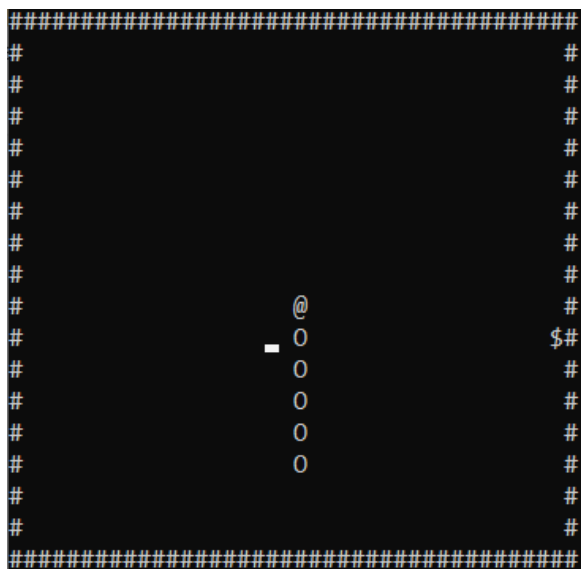
4.Руководство пользователя

Игра “змейка” – 2D с элементами пользовательского интерфейса.

При загрузке откроется меню , в котором вам нужно ввести свое имя .

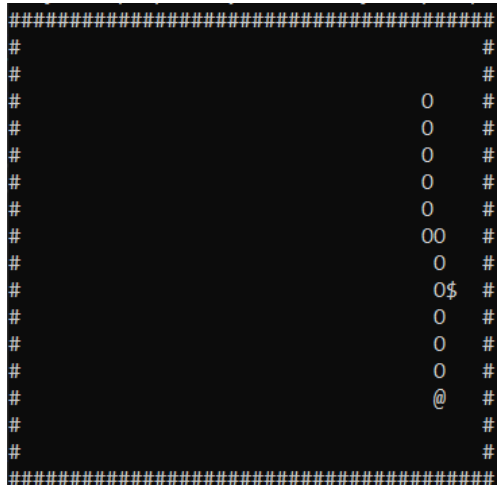


После ввода имени загрузится поле для игры .



При нажатие клавиши “W” змейка начнет двигаться вверх , при нажатии “D” змейка начнет двигаться вправо , при нажатии клавиши “S” змейка начнет двигаться вниз , при нажатии клавиши “A” змейка начнет двигаться влево. Если змейка врежется в границу или же в свой хвост , тогда игра будет окончена .Также если змейка двигается в одном направлении , вы не можете направить ее в противоположное направление. Когда змейка съедает

фрукта , тогда она становится больше .



Если вы хотите выйти из игры заранее, то можете нажать клавишу “L”

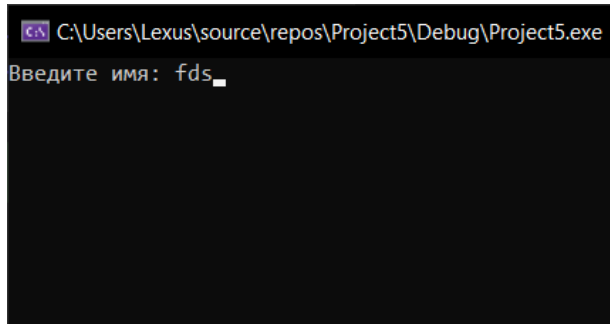
5. Разработка функции на языке Ассемблер.

В своей работе я использовал ассемблер для определения середины поля , чтобы расположить там голову змейки

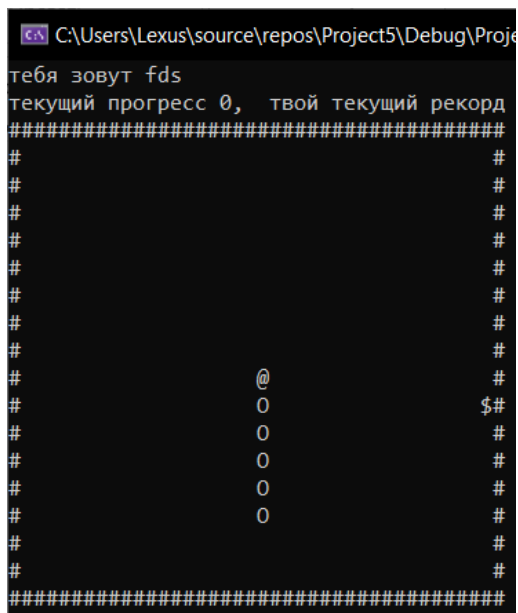
```
_asm {  
    mov eax, h  
    mov ebx, 2  
    sub eax, ebx  
    cdq  
    idiv ebx  
    mov snake[0].x, eax  
    mov eax, w  
    cdq  
    idiv ebx  
    mov snake[0].y, eax  
}
```

6.Тестирование программы

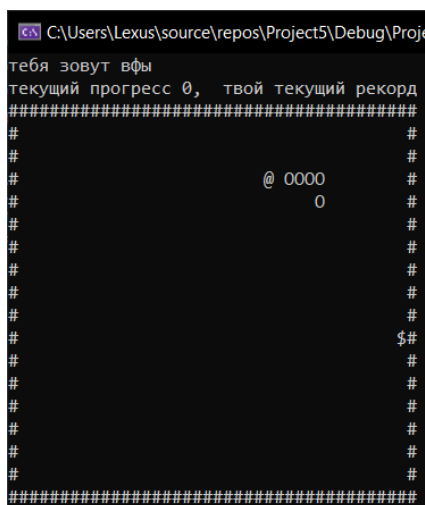
1. Запуск программы



2. После ввода имени запускается окно 40x20 с расположением змейки по центру



3. Все клавиши работает нормально, включая клавишу выхода "L"



Содержание:

- 1.Титульный лист(стр.1)
- 2.Постановка задачи(стр.2)
- 3.Описание структуры программы(стр.3-4)
- 4.Описание функционала программы(стр.4-5)
- 5.Руководство пользователя(стр.5-6)
- 6.Вставка на ассемблере(стр.6)
- 7.Тестирование (стр.7-8)
- 8.Заключение(стр.8)

Листинг программы

Файл “Project5”

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <iostream>
5  #include <conio.h>
6  #include <Windows.h>
7  #include <time.h>
8  #include <malloc.h>
9  #include <C:\Users\Lexus\source\repos\Project5\Project5\Header.h>
10 #define h 20
11 #define w 40
12
13 char apls = '$'; //скин яблока
14 char golova = '0'; //скин головы змеи
15 char telo = '0'; //скин хвоста змеи
16 char karta = '#'; //скин границ карты
17 char name[30], rec[40];
18 int score = 0, juij = 0, i = 0, j = 0, akkload = 0, oldrec = 0, n = 0, nachlen = 5, record = 0, rec_m = 0;
19 char pole[h][w];
20 short int crtime = 0; //время движения
21 bool gameover = false;
22 char text[60];
23 Enum pere {
24     stop = 0,
25     up,
26     down,
27     left,
28     right
29 };
30 enum pere qas;
31 typedef struct snst {
32     int x;
33     int y;
34     char jb = telo;
35 } snstt;
36
37 typedef struct aps {
38     short int x;
```

```

37
38 typedef struct aps {
39     short int x;
40     short int y;
41     bool gen;
42     char jb = apls;
43 }apss;
44 apss apl;
45 snstt snake[1600];
46 snstt bfs;
47 void clear(int x, int y)
48 {
49     COORD coord;
50     coord.X = x;
51     coord.Y = y;
52     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
53 }
54 void game()
55 {
56     for (int i = 0; i < h - 2; i++)
57     {
58         for (int j = 0; j < w; j++)
59         {
60             pole[i][j] = karta;
61         }
62     }
63     for (int i = 1; i < h - 3; i++)
64     {
65         for (int j = 1; j < w - 1; j++)
66         {
67             pole[i][j] = ' ';
68         }
69     }
70 }
71 void gen_food()
72 {
73     if (apl.gen == false)
74     {
75         apl.x = 1 + rand() % (h - 4);

```

```

71 void gen_food()
72 {
73     if (apl.gen == false)
74     {
75         apl.x = 1 + rand() % (h - 4);
76         apl.y = 1 + rand() % (w - 2);
77         pole[apl.x][apl.y] = apl.jb;
78         apl.gen = true;
79         for (int n = 0; n < juij; n++)
80         {
81             if ((apl.x == snake[n].x) and (apl.y == snake[n].y)) { apl.gen = false; }
82         }
83     }
84     else
85     {
86         pole[apl.x][apl.y] = apl.jb;
87     }
88 }
89 //отрисовка змейки\
90 void createsnake()
91 {
92     for (int i = 0; i <= juij; i++)
93     {
94         if ((snake[i].x != 0) and (snake[i].y != 0))
95         {
96             pole[snake[i].x][snake[i].y] = snake[i].jb;
97         }
98     }
99 }
100 //движение змейки\
101 void xvost_ypr()
102 {
103     crtime++;
104     if (crtime == 2)
105     {
106         if (qas == up) {
107             if ((pole[snake[0].x - 1][snake[0].y] != karta) and (pole[snake[0].x - 1][snake[0].y] != telo))
108             {
109                 bfs.x = snake[0].x;

```

```

107     if ((pole[snake[0].x - 1][snake[0].y] != karta) and (pole[snake[0].x - 1][snake[0].y] != telo))
108     {
109         bfs.x = snake[0].x;
110         bfs.y = snake[0].y;
111         snake[0].x = snake[0].x - 1;
112     }
113     else { gameover = true; }
114 }
115 if (qas == down)
116 {
117     if ((pole[snake[0].x + 1][snake[0].y] != karta) and (pole[snake[0].x + 1][snake[0].y] != telo))
118     {
119         bfs.x = snake[0].x;
120         bfs.y = snake[0].y;
121         snake[0].x = snake[0].x + 1;
122     }
123     else { gameover = true; }
124 }
125 if (qas == left)
126 {
127     if ((pole[snake[0].x][snake[0].y - 1] != karta) and (pole[snake[0].x][snake[0].y - 1] != telo))
128     {
129         bfs.x = snake[0].x;
130         bfs.y = snake[0].y;
131         snake[0].y = snake[0].y - 1;
132     }
133     else { gameover = true; }
134 }
135 if (qas == right)
136 {
137     if ((pole[snake[0].x][snake[0].y + 1] != karta) and (pole[snake[0].x][snake[0].y + 1] != telo))
138     {
139         bfs.x = snake[0].x;
140         bfs.y = snake[0].y;
141         snake[0].y = snake[0].y + 1;
142     }
143     else { gameover = true; }
144 }
145 crtime = 0;

```

```

143     crtime = 0;
144 }
145 }
146 void printpgame()
147 {
148     printf_s(_Format: "тебя зовут %s\n", name);
149     printf(_Format: "текущий прогресс %d, твой текущий рекорд %d \n", score, record);
150     for (int i = 0; i < h - 2; i++)
151     {
152         for (int j = 0; j < w; j++)
153         {
154             printf(_Format: "%c", pole[i][j]);
155         }
156         if (i != h - 3) { printf(_Format: "\n"); }
157     }
158 }
159 void xvost()
160 {
161     if (crtime == 1)
162     {
163         for (int i = juij; i >= 1; i--)
164         {
165             if (i > 1)
166             {
167                 snake[i].x = snake[i - 1].x;
168                 snake[i].y = snake[i - 1].y;
169             }
170             if (i == 1)
171             {
172                 snake[i].x = bfs.x;
173                 snake[i].y = bfs.y;
174             }
175         }
176     }
177 }
178 void food()
179 {
180     if ((snake[0].x == apl.x) and (snake[0].y == apl.y))
181     {

```

```

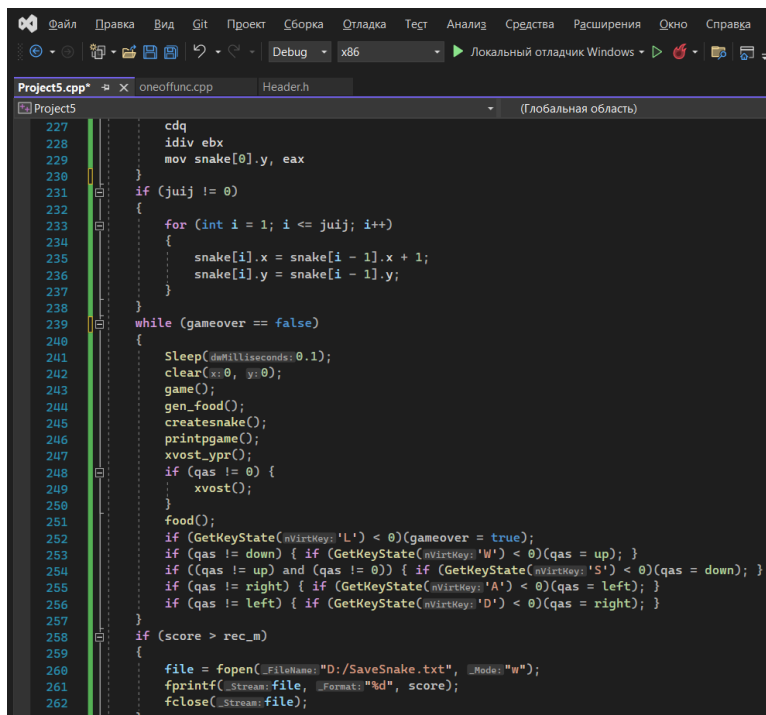
179 {
180     if ((snake[0].x == apl.x) and (snake[0].y == apl.y))
181     {
182         apl.gen = false;
183         score++;
184         if (score > record)
185         {
186             record = score;
187             if (oldrec == record)
188             {
189                 oldrec = score;
190             }
191         }
192         juij++;
193     }
194 }
195 int main()
196 {
197     system(_Command: "chcp 1251");
198     system(_Command: "cls");
199     printf(_Format: "Введите имя: ");
200     scanf_s(_Format: "%s", name, 30);
201     system(_Command: "cls");
202     FILE* file;
203     if ((file = fopen(_FileName: "D:/SaveSnake.txt", _Mode: "r")) == NULL) {
204         file = fopen(_FileName: "D:/SaveSnake.txt", _Mode: "w");
205         fprintf(_Stream: file, _Format: "0");
206         fclose(_Stream: file);
207     }
208     if ((file = fopen(_FileName: "D:/SaveSnake.txt", _Mode: "r")) != NULL)
209     {
210         file = fopen(_FileName: "D:/SaveSnake.txt", _Mode: "r");
211         fgets(_Buffer: rec, _MaxCount: 40, _Stream: file);
212         record = atoi(_String: rec);
213         rec_m = record;
214     }
215     hidecursor();
216     juij = nachlen;
217     apl.gen = false;

```

```

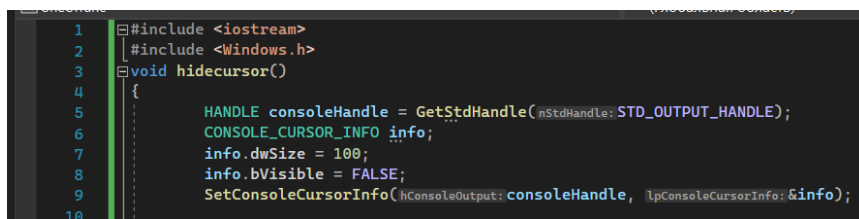
214 }
215 hidecursor();
216 juij = nachlen;
217 apl.gen = false;
218 snake[0].jb = golova;
219 _asm {
220     mov eax, h
221     mov ebx, 2
222     sub eax, ebx
223     cdq
224     idiv ebx
225     mov snake[0].x, eax
226     mov eax, w
227     cdq
228     idiv ebx
229     mov snake[0].y, eax
230 }
231 if (juij != 0)
232 {
233     for (int i = 1; i <= juij; i++)
234     {
235         snake[i].x = snake[i - 1].x + 1;
236         snake[i].y = snake[i - 1].y;
237     }
238 }
239 while (gameover == false)
240 {
241     Sleep(dwMilliseconds: 0.1);
242     clear(x: 0, y: 0);
243     game();
244     gen_food();
245     createsnake();
246     printpgame();
247     xvost_ypr();
248     if (qas != 0) {
249         xvost();
250     }
251     food();
252     if (GetKeyState(nVirtKey: 'L') < 0)(gameover = true);

```



```
227      cdq
228      idiv ebx
229      mov snake[0].y, eax
230  }
231  if (juij != 0)
232  {
233      for (int i = 1; i <= juij; i++)
234      {
235          snake[i].x = snake[i - 1].x + 1;
236          snake[i].y = snake[i - 1].y;
237      }
238  }
239  while (gameover == false)
240  {
241      Sleep(dwMilliseconds:0.1);
242      clear(x:0, y:0);
243      game();
244      gen_food();
245      createsnake();
246      printgame();
247      xvost_ypr();
248      if (qas != 0) {
249          xvost();
250      }
251      food();
252      if (GetKeyState(nVirtKeys: 'L') < 0)(gameover = true);
253      if (qas != down) { if (GetKeyState(nVirtKeys: 'W') < 0)(qas = up); }
254      if ((qas != up) and (qas != 0)) { if (GetKeyState(nVirtKeys: 'S') < 0)(qas = down); }
255      if (qas != right) { if (GetKeyState(nVirtKeys: 'A') < 0)(qas = left); }
256      if (qas != left) { if (GetKeyState(nVirtKeys: 'D') < 0)(qas = right); }
257  }
258  if (score > rec_m)
259  {
260      file = fopen(_FileName: "D:/SaveSnake.txt", _Mode: "w");
261      fprintf(_Stream: file, _Format: "%d", score);
262      fclose(_Stream: file);
263  }
```

Файл “oneoffunc”



```
1  #include <iostream>
2  #include <Windows.h>
3  void hidecursor()
4  {
5      HANDLE consoleHandle = GetStdHandle(STD_OUTPUT_HANDLE);
6      CONSOLE_CURSOR_INFO info;
7      info.dwSize = 100;
8      info.bVisible = FALSE;
9      SetConsoleCursorInfo(consoleHandle, &info);
10 }
```

