Отчет по курсовой работе  
Задание 1

# Описание

Программа по тестовому заданию в цикле определяет нажатие клавиш управления змейкой и выводит наименование нажатой клавиши на экран. Выход из цикла выполняется по игровой клавише «Выход».

# Рабочий экран



Рисунок . Клавиша неопределена



Рисунок . Клавиша определена

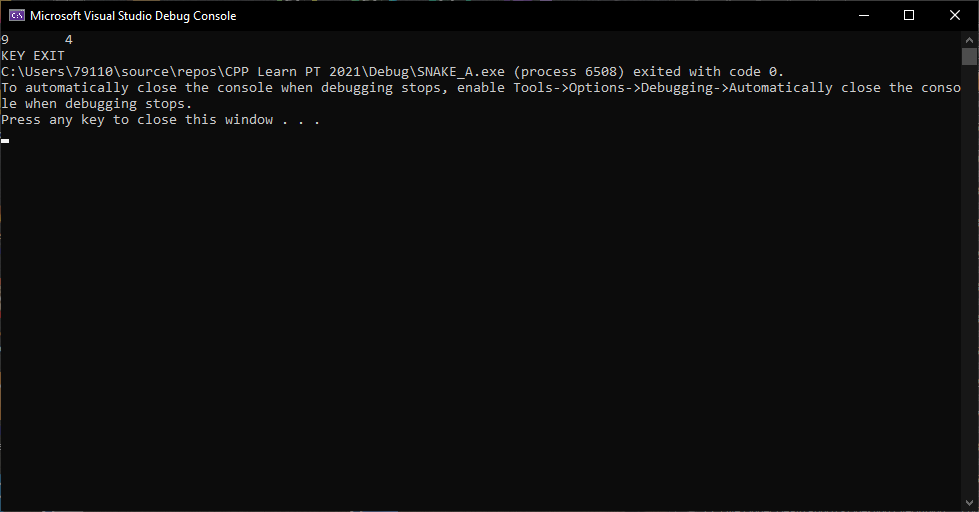


Рисунок . Клавиша выхода

# Состав программы

Проект содержит 2 файла хедеров и 2 файла кода:  
inputs.h  
test\_inputs.h  
inputs.cpp  
test\_inputs.cpp

## inputs.h

#pragma once

#if defined \_INPUTS\_SET

#define \_INPUTS\_SET

#endif //

#define ESC\_KEY 27

enum Actions

{

KEY\_UP,

KEY\_DN,

KEY\_RT,

KEY\_LT,

EXIT,

UNDEF,

OP1,

OP2,

OP3

};

int read\_input();

## inputs.cpp

#include <conio.h>

#include <ctype.h>

#include "inputs.h"

struct KeyMapItem

{

unsigned int key;

char action;

};

static KeyMapItem keyMap[]

{

{75, KEY\_LT},

{77, KEY\_RT},

{72, KEY\_UP},

{80, KEY\_DN},

{'A', KEY\_LT},

{'D', KEY\_RT},

{'W', KEY\_UP},

{'S', KEY\_DN},

{'Ф', KEY\_LT},

{'В', KEY\_RT},

{'Ц', KEY\_UP},

{'Ы', KEY\_DN},

{'0', EXIT},

{ESC\_KEY, EXIT},

{'1', OP1},

{'2', OP2},

{'3', OP3}

};

int read\_input()

{

int mapSize = sizeof(keyMap) / sizeof(keyMap[0]);

int ch;

ch = \_getch();

ch = toupper(ch);

#if \_DEBUG

\_putch(ch);

\_putch('\r'); // Carriage return

\_putch('\n'); // Line feed

#endif

char action = UNDEF;

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

{

if (ch == keyMap[i].key)

{

action = keyMap[i].action;

break;

}

}

return action;

}

## test\_inputs.h

#pragma once

int test\_inputs();

## test\_inputs.cpp

#include <iostream>

#include <conio.h>

//#include <stdio.h>

//#include <string.h>

//#include <cstdlib>

#include "test\_screen.h"

#include "inputs.h"

void do\_up()

{

std::cout << "KEY UP";

}

void do\_dn()

{

std::cout << "KEY DN";

}

void do\_rt()

{

std::cout << "KEY RT";

}

void do\_lt()

{

std::cout << "KEY LT";

}

void do\_xt()

{

std::cout << "KEY EXIT";

std::exit(EXIT\_SUCCESS);

}

void do\_ud()

{

std::cout << "KEY UNDEFINED";

}

void do\_op1()

{

//test\_screen();

std::cout << "KEY OP1\t";

move\_snake\_test();

}

void do\_op2()

{

//test\_screen();

std::cout << "KEY OP2\t";

move\_snake\_test1();

}

void do\_op3()

{

//test\_screen();

std::cout << "KEY OP2\t";

move\_snake\_test2();

}

using actFx = void (\*)();

struct ActMapItem

{

char key;

actFx action;

};

static ActMapItem actMap[]

{

{KEY\_UP, do\_up},

{KEY\_DN, do\_dn},

{KEY\_LT, do\_lt},

{KEY\_RT, do\_rt},

{EXIT, do\_xt},

{UNDEF, do\_ud},

{OP1, do\_op1},

{OP2, do\_op2},

{OP3, do\_op3}

};

int test\_inputs()

{

int actMapSize = sizeof(actMap) / sizeof(actMap[0]);

char ch;

void (\* pAction)();

while (true)

{

ch = read\_input();

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

{

if (ch == actMap[i].key)

{

system("CLS");

#if defined \_DEBUG

std::cout << actMapSize << '\t' << i << '\n';

#endif

pAction = actMap[i].action;

pAction();

break;

}

}

}

return 0;

}