|  |  |
| --- | --- |
| Министерство науки и высшего образования Российской Федерации  Федеральное государственное автономное образовательное  учреждение высшего образования  «Южно-Уральский государственный университет  (национальный исследовательский университет)»  Институт естественных и точных наук  Кафедра прикладной математики и программирования | |
| Пасьянс «Маджонг» | |
| Пояснительная записка к курсовой работе  по дисциплине «Методы программирования»  ЮУрГУ–010302.2023.070.ПЗ КР | |
|  | Автор работы,  студент группы ЕТ-112  \_\_\_\_\_\_\_\_\_\_ / Т.В. Щипков  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  | Руководитель работы,  старший преподаватель  \_\_\_\_\_\_\_\_\_\_ / М.Ю. Сартасова  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  | Работа защищена с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г. |
| Челябинск 2023 | |

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет   
(национальный исследовательский университет)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

Направление Прикладная математика и информатика

УТВЕРЖДАЮ

Заведующий кафедрой ПМиП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А.Замышляева

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**на курсовую работу студента**

\_\_\_\_\_\_\_\_*Щипкова Т.В.*\_\_\_\_\_\_\_\_\_

Группа \_\_*ЕТ-112*\_\_\_\_

1. Дисциплина  *Методы программирования*

2. Тема работы \_\_\_*Пасьянс «Маджонг»* \_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Срок сдачи студентом законченной работы *10 июня 2023 г.*

4. Перечень вопросов, подлежащих разработке

1. постановка задачи и алгоритм программы;
2. реализация программы на языке С++
3. оформление программной документации (руководство пользователя, листинг кода) и отчета по курсовой работе
4. презентация проектных решений для защиты КР (постановка задачи, разработка алгоритма, особенности реализации)

5. Календарный план

|  |  |  |
| --- | --- | --- |
| **Наименование разделов**  **(этапов) курсовой работы** | **Срок выполнения**  **разделов (этапов) работы** | **Отметка**  **о выполнении**  **руководителя** |
| постановка задачи | 24.02.2023-08.03.2023 |  |
| разработка алгоритма | 09.03.2023-05.04.2023 |  |
| реализация программы | 06.04.2023-26.04.2023 |  |
| тестирование программы, улучшение и исправление ошибок | 27.04.2023-17.05.2023 |  |
| оформление программной документации и отчета по курсовой работе | 18.05.2023-31.05.2023 |  |
| защита курсовой работы | 01.06.2023-14.06.2023 |  |

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (расшифровка)

АННОТАЦИЯ

Щипков Т.В. Пасьянс «Маджонг». – Челябинск: ЮУрГУ, ЕТ-112, 2023. – 27с., 8 ил., библиографический список – 2 наим., 1 прил.

В курсовой работе описывается разработка пасьянса «Маджонг» на языке программирования С++.

Целью курсовой работы является получение навыков создания программного обеспечения на основе структурного подхода при использовании интегрированной среды разработки.

В процессе работы были выполнены все этапы разработки программы: постановка задачи, проектирование программы, включающее нисходящее проектирование и структурное программирование, реализация и отладка программы.

Пояснительная записка содержит результаты выполнения этих этапов, разработанные структуры данных и схемы алгоритмов, использованные математические модели, а также руководство пользователя программы, включающее примеры интерфейса.

В результате работы была разработан пасьянс «Маджонг», код которого приводится в приложении.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc136711854)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc136711855)

[2 РАЗРАБОТКА АЛГОРИТМА 7](#_Toc136711856)

[3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 9](#_Toc136711857)

[ЗАКЛЮЧЕНИЕ 12](#_Toc136711858)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 13](#_Toc136711859)

[ПРИЛОЖЕНИЕ а 14](#_Toc136711860)

# ВВЕДЕНИЕ

**Актуальность темы**. Четкое соблюдение порядка этапов разработки программного обеспечения, понимание целей каждого из этих этапов, грамотное применение технологий и сред разработки обеспечивает реализацию сложных программных систем качественно и в срок.

**Цель работы** – разработать игру пасьянс «Маджонг»

**Задачи работы**:

– научиться корректно осуществлять постановку задачи;

– изучить методики проектирования программ, включая нисходящее проектирование и структурное программирование;

– научиться разрабатывать и описывать алгоритмы на основе структурного подхода с применением метода пошаговой детализации и стандартных графических обозначений;

– научиться выполнять разработку и отладку программы для ее решения;

– получить навыки работы с различными средами программирования и прикладными библиотеками;

– овладеть способами планирования, организации, самоконтроля и самооценки деятельности;

– осуществлять целенаправленный поиск информации в сети Интернет по теории и практике программирования, математическим алгоритмам, библиотекам для разработки консольных и графических программ.

**Объект работы** – программа для игры в пасьянс «Маджонг».

**Предмет работы** – применение технологий разработки программного обеспечения на основе структурного подхода и языка С++ для разработки программы.

**Результаты работы** можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика».

## 1 ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать программу, реализующую игру «Маджонг». Для разработки необходимо использовать язык программирования C++[1] и графическую библиотеку winBGIm[2]. Среда разработки: MinIDE.

После запуска программы на экране должно появится главное окно, представленное на рисунке 1.



Рисунок 1. – Главное меню

При нажатии пользователем кнопки «Новая игра» открывается выбор игрового поля (построения фигур). После предварительного выбора, пользователь попадает на игровое поле (рисунок 2).

Рисунок 2. – Игровое поле

На игровом поле есть кнопка «Меню» - открывает соответствующий список с опциями «Продолжить», «Поменять раздачу», «Выход».

Также есть характерная кнопка «Перемешать» - произойдет перемешивание фигур, если больше мы не сможет подобрать пары для фишек.

Кнопка с глазом показывает какие фишки закрытые, путем затемнения них.

Лампочка показывает, какие фишки образуют пару если пользователь не может найти её.

При открытии меню «Правила», открывается список правил:

* Пользователь выбирает фишку и должен подобрать к ней пару;
* Некоторые фишки нельзя выбрать, поэтому пользователю можно выбрать самые левые и правые фишки, а также фишки слоем выше;
* Такими действиями мы должны постепенно освободить стол от фишек путем подбор пар;

Выбор пункта «О программе» описывает название программы, создателя программы и т. д.

При выборе опции «Выход» пользователь выходит из программы.

## 2 РАЗРАБОТКА АЛГОРИТМА

В программе используется таблица для игрового поля, которая будет отображать фишки. Для отображения таблицы на экране будем использовать трехмерный массив Pole[13][7][6], каждый элемент которой имеет значение в соответствии с этой формулой.

Изначально, массив изначально будет заполнен числами -1 и 0 – для обозначения куда мы можем вставлять фишки. Для показа используем пример с трехмерным массивом Pole[3][3][3]. Для него будет такая раскладка фишек в виде «Пирамиды».

Создаем вектор пар layout, который берет значение из массива структуры tiles, где:

Далее мы через функцию random\_shuffle перемешиваем фишки в вектор пар layout. Данный вектор предназначен для раскладки стола, чтобы потом мы напрямую заносили значения layout в массив Pole для составления стола игры. После мы заносим значение id отсортированного вектора в массив Pole;

Перед началом игры создается счетчик CON\_TILES равный 144. В процессе игры пользователь выбирает пару фишек, и вместо их id элемент приравнивается к -1, из layout удаляются эти фишки и также удаляются с поля. Также счетчик CON\_TILES уменьшается на 2 при каждом подборе пары.

Проверка на победу осуществляется через проверку на количество доступных пар фишек. Если CON\_TILES равен нулю, то мы выиграли партию. Если нет доступных пар, то игра заканчивается. Либо пользователь может попробовать выиграв, повторно перемешав фишки, находящиеся на столе.

Схема алгоритма игры представлена на рисунке 3. Также на рисунке 4 представлен алгоритм меню.

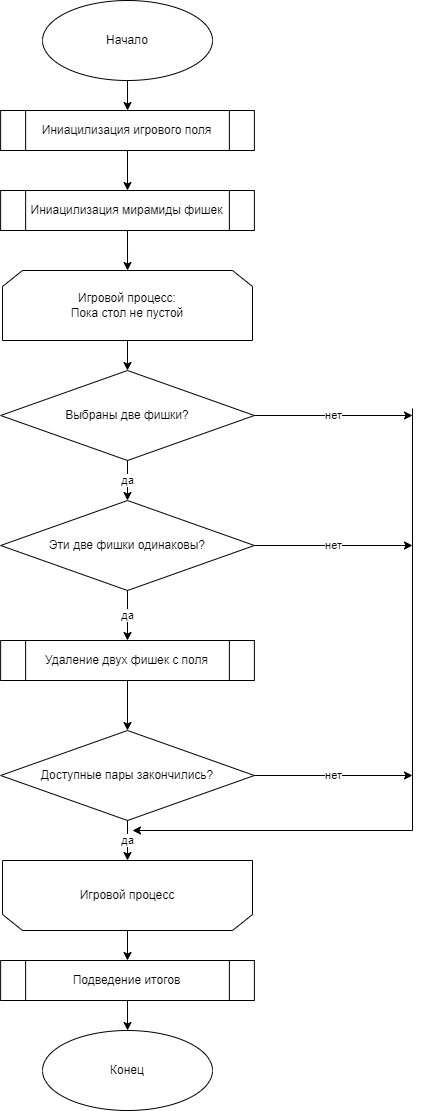


Рисунок 3 – Основной алгоритм программы

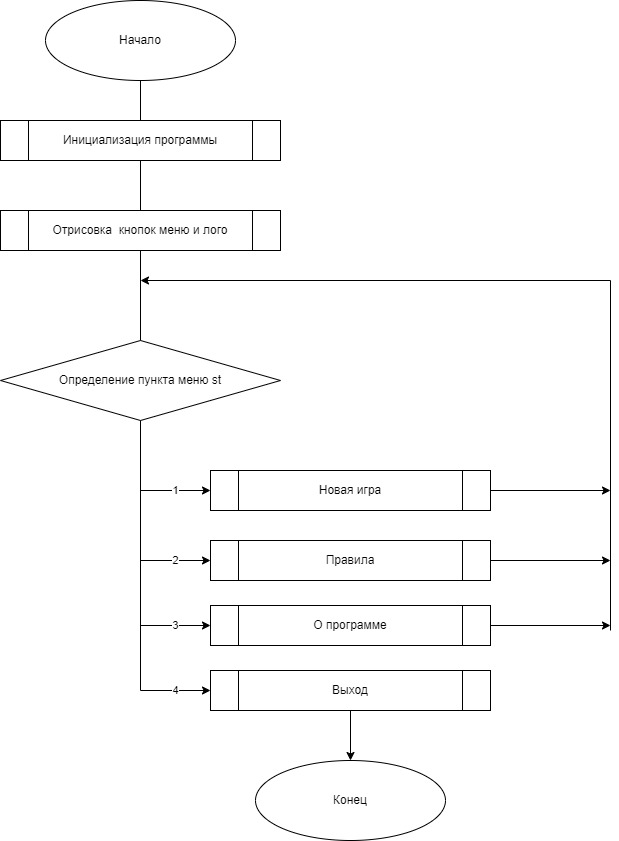


Рисунок 4 – Алгоритм меню

## 3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске программы открывается главное меню игры (рисунок 5).

В данном меню мы имеем 4 опции. Нажав на «Играть» (рисунок 6) пользователь начинает игру.

Сверху пользователь видит основную информацию об текущей игре – текущее время прохождения игры, оставшееся количество фишек и сколько доступных пар находится на игровом поле. Кнопка «Выход» в основной игре позволяет выйти в главное меню, и, если пользователь нажмет «Играть», ему будет предоставлен выбор: начать новую игру или продолжить старую.

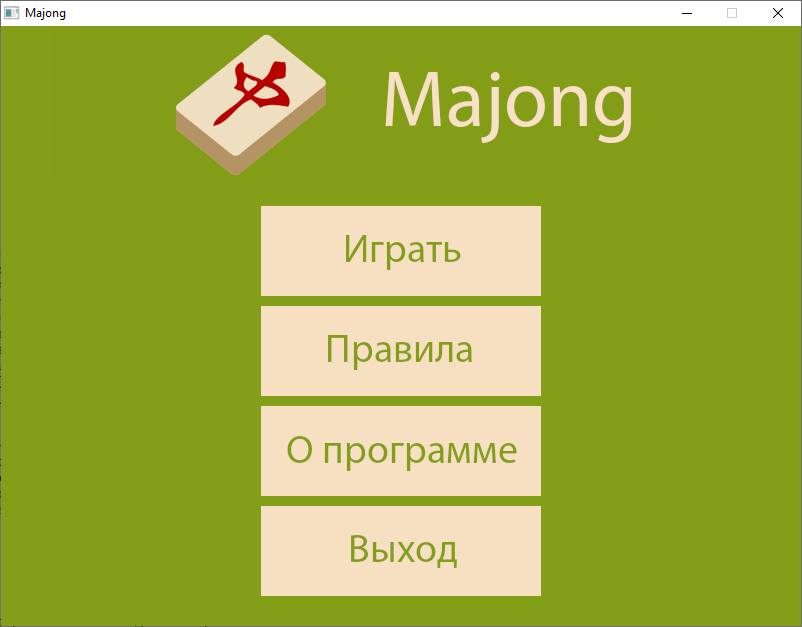


Рисунок 5 – Окно меню

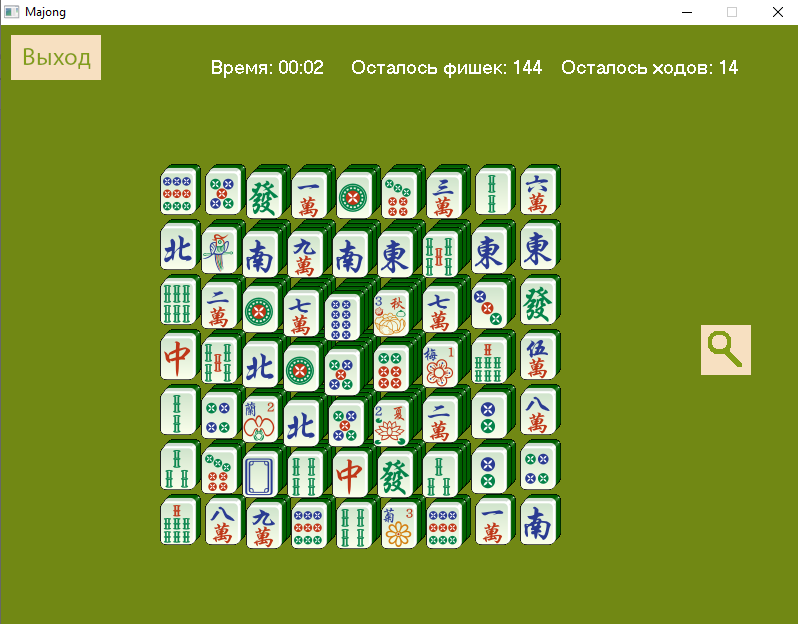


Рисунок 6 – Окно игры

В середине находится поле с фишками, которое нужно расчистить. Фишку можно выделить нажатием левой кнопки мыши, при этом она выделяется белой рамкой. Повторным нажатием на другую фишку пользователь либо убирает пару фишек, либо выбирает снова. Кнопка с лупой показывает для пользователя пару фишек, выделяя их желтой рамкой. При отсутствии доступных пар выводится окно о проигрыше с двумя выборами – продолжить игру, перемешав фишки на поле, либо выйти в главное меню. Когда пользователь отчистил поле от фишек, выводится окно о выигрыше и время прохождения данной игры с последующим выходом в главное меню.

Кнопка «Правила» отвечает за вывод правил игры (рисунок 7).

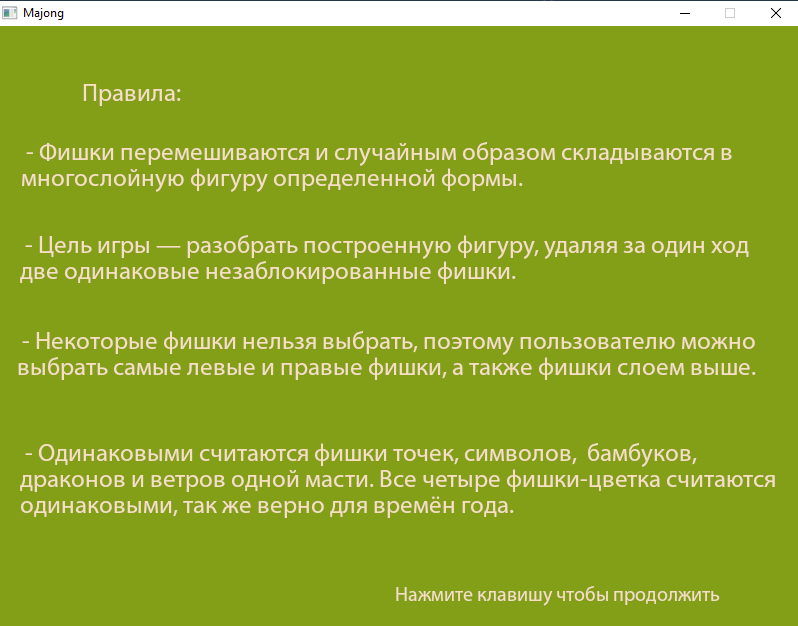


Рисунок 7 – Окно правил

«О программе» выводит основную информацию о создателе игры (рисунок 8).

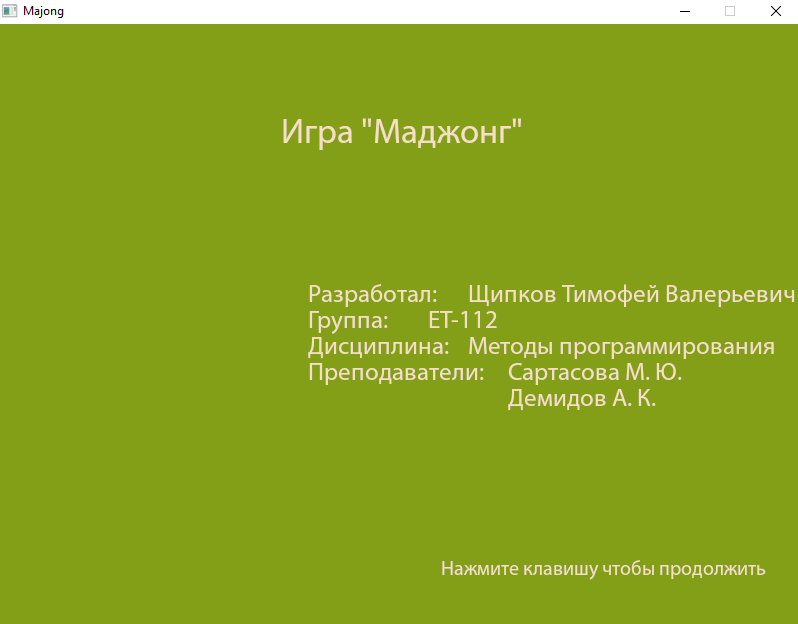


Рисунок 8 – Окно программы

Кнопка «Выход» обеспечивает выход из программы.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были поставлены требования к программе, затем были разработаны необходимые математические модели, определены и детализированы структуры данных и алгоритмы. После завершения проектирования алгоритмы были реализованы на языке С++. Разработанный код был проверен и были внесены необходимые исправления. Для программы было разработано руководство пользователя. Таким образом, цель работы была достигнута, задачи – решены.

Результаты работы можно использовать в процессе последующего обучения в форме навыков практического применения структурного программирования для разработки сложных программных систем, понимания порядка этапов разработки программного обеспечения и достигаемых на каждом этапе результатов.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Подбельский, В. В. Язык Си++. Учебное пособие. 5-е издание.

2 Графическая библиотека WinBGIm. – URL: <https://ipc.susu.ru/20786.html>

# ПРИЛОЖЕНИЕ а

А.1 Файл main.cpp

#include "menu.h"

int main()

{

initwindow(width, height, "Majong", 200, 200, true); //обьявление графического окна

begin(); // инициализация и работа с меню (определена в menu.cpp)

return 0;

}

#endif

А.2 Файл menu.h

#ifndef \_MENU\_

#define \_MENU\_

#include "graphics.h"

#define ELVOLT COLOR(131, 158, 23) //цвет, используемый для фона

// длина и ширина графического окна

#define width 800

#define height 600

typedef struct button { // обьявление структуры кнопок

int x, y, dx, dy;

IMAGE \* bmp;

} Button;

// прототипы функций

void begin(); // приготовление меню и переназначение на окна

void init\_menu(); //инициализация меню

int menu(int st); // функция для определения куда нажал пользователь

void drawmenu(); // отрисовка лого и кнопок меню

void rules\_about(int st); // функция для отрисовки правил/о программе

void close\_game(); //закрытие графического окна

void setVSPage();

void setACPage();

void continue\_game();

#endif

A.3 Файл menu.cpp

#include "menu.h"

#include "new\_game.h"

#include "graphics.h"

button but[4]; //обьявление массива кнопок для меню

button logo; // лого игры для меню

button rulAbt; // кнопка для выведения окна "Правила"/"О программе"

const int buttonW = 280, buttonH = 90, mid\_width = width / 2, mid\_height = height / 2; //размеры кнопок меню

int currentPage = -1; // для новых окон и анимации

bool contGame = false;

void begin()// приготовление меню и переназначение на окна

{

int st = 1; //статус - какой пункт меню будет нажат

init\_menu(); //подготовка и отрисовка меню

do{

st = menu(st);

switch(st){ // переназначение на окно которое выбрал пользователь

case 1:

if (contGame) continue\_game();

else new\_game();

break; // Новая игра

default: //Правила/О программе

if (st != 4) rules\_about(st);

break;

}

} while(st != 4);

close\_game(); //закрытие графического окна

}

void continue\_game()

{

button res[3];

char s[25];

setVSPage();

clearviewport();

for(int i = 0; i < 3; i++) // указ координат кнопок и их адрес

{

sprintf(s, ".//MENU\_STUFF/restart%d.bmp", i);

if(i != 0)

{

res[i].dx = 280; res[i].dy = 90;

res[i].x = 100 + res[i].dx\*(i - 1) + (i - 1)\*40; res[i].y = 450;

}

res[i].bmp = loadBMP(s);

}

for(int i = 0; i < 3; i++) // вывод экрана проигрыша и кнопок для выбора

{

if (i == 0) putimage(0, 0, res[i].bmp, COPY\_PUT);

else putimage(res[i].x, res[i].y, res[i].bmp, COPY\_PUT);

}

setACPage();

int flag = 0, x, y, st = 0; //какая кнопка была нажата

do {

while(mousebuttons() != 1){

x = mousex();

y = mousey();}

for(int i = 1; i < 3; i++){

if (x >= res[i].x && x <= res[i].x + res[i].dx && y >= res[i].y && y <= res[i].y + res[i].dy)

{flag = 1; st = i+1; break; }

}

}while(!flag);

while(mousebuttons()==1);

if(st == 2) {contGame = true;}

else contGame = false;

new\_game();

}

void init\_menu(){

char file[30]; //массив для обозначения пути файла кнопки

for(int i=0; i <4; i++){ // присваивание координат кнопкам а также указание пути

but[i].dx = 280; but[i].dy = 90;

but[i].x = (width / 2) - (buttonW / 2); but[i].y = 180 + i \* (but[i].dy+10);

sprintf(file,".//MENU\_STUFF/menu%d.bmp", i+1);

but[i].bmp = loadBMP(file);

}

// тоже самое отдельно для лого

logo.x = 50; logo.y = 0;

logo.bmp = loadBMP(".//MENU\_STUFF/menu\_majong.gif");

}

void drawmenu() // отрисовка лого и кнопок меню

{

setbkcolor(ELVOLT); // установка цвета фона

setVSPage();

clearviewport();

putimage(logo.x, logo.y, logo.bmp, COPY\_PUT);

for(int i=0; i <4; i++){ //вставка на экран лого и кнопок

putimage(but[i].x , but[i].y, but[i].bmp, COPY\_PUT);

}

setACPage();

}

int menu(int st)

{

int flag = 0, x, y;

drawmenu(); //отрисовка меню

do {

while(mousebuttons() != 1){ //нахождение координат куда нажал пользователь а также определение статуса

x = mousex();

y = mousey();}

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

if (x >= but[i].x && x <= but[i].x + but[i].dx && y >= but[i].y && y <= but[i].y + but[i].dy) //проверка что клик находится в границах кнопки

{flag = 1; st = i+1; break; }

}

}while(!flag);

return st;

}

void rules\_about(int st)

{

setVSPage();

clearviewport();

if (st == 2) rulAbt.bmp = loadBMP(".//MENU\_STUFF/rules.bmp"); //проверка какой файл надо вывести на экран

else rulAbt.bmp = loadBMP(".//MENU\_STUFF/about.bmp");

putimage(0, 0, rulAbt.bmp, COPY\_PUT); //вывод

setACPage();

while(kbhit()) getch();

getch(); //ожидание нажатия кнопки

}

void close\_game(){ closegraph(); } // закрытие графического окна

void setVSPage()

{

if(currentPage == 3) currentPage = 0;

currentPage++;

setactivepage(currentPage);

}

void setACPage(){ setvisualpage(currentPage); }

A.4 Файл new\_game.h

#ifndef \_GAME\_

#define \_GAME\_

#define AVOCADO COLOR(113, 136, 20)

#define BEIGE COLOR(247, 224, 193)

typedef struct TILE { //структура фишки

char name[20];

int id, count, x, y, i, j, k; //id фишки, её количество в игре, координата по x и y, координаты фишки в массиве

IMAGE \* bmp;

}Tile;

// прототипы функций

void new\_game(); //отрисовка массива и движок игры

void draw\_pole(); //отрисовывает фишки на поле, а также сколько осталось и сколько пар доступно

void init\_game(); // инициализация библиотеки и раскладки

void maj\_init(); //предварительное создание поля и его заполнение

void find\_tiles(); // нахождение фишек если пользователь их не видит

bool definition\_XY(int \*i, int \*y, int \*k); // определение координат в массиве

void core\_game(); // основной процесс игры

bool click(int \*i, int \*j); // определение какую фишку выбрал пользователь

void delete\_pair(TILE \*tile1, TILE \*tile2); // удаление фишек

bool is\_avalible(TILE\* tile1); //доступна ли фишка

bool is\_season(int tile1, int tile2); // проверка сезонная ли фишка

void acc\_avl(); //пересчет доступных пар фишек

void mix\_at\_end(); // перемешивание при отсутсвующих фишках

void border(TILE \*tile); // границы при нажатии на фишку(не работает с swapbuffers())

void stopwatch(); // реализация секундомера

void printSW(); // вывод секундомера

void end(); //окно при закончившихся доступных фишек

void victory(); // окно победы с выходом в главное меню

#endif

A.5 Файл new\_game.cpp

#include<iostream>

#include<vector>

#include<random>

#include <algorithm>

#include <utility>

#include <cmath>

#include <ctime>

#include "menu.h"

#include "new\_game.h"

#include "graphics.h"

using namespace std;

const int le = 9, wi = 7, he = 5; // размеры массива пирамиды

TILE Pole[le][wi][he]; // обьявление трехмерного массива

const int tileW = 45, tileH = 55; //размеры фишки в пикселях

vector <pair<int, int>> layout; //раскладка фишек

vector <TILE> for\_find; // промежуточный вектор для поиска find\_tiles

pair<TILE,TILE> founds; // найденная пара для find\_tiles

// для рандомизации раскладки

auto rd = random\_device {};

auto rng = default\_random\_engine {rd()};

//

int begOfX = floor((width - (tileW \* le)) / 2) - 50; //начальная координата по X для вывода всей пирамиды

int begOfY = floor((height - (tileH \* wi)) / 2) + 30; //начальная координата по Y для вывода всей пирамиды

int pairAVL, CON\_TILES; //количество доступных фишек, количество всех фишек

button lose, win, gMenu, findTiles; //окна для вывода проигрыша или выигрыше

TILE library[42]; //библиотка для фишек

bool pereB; //для включение/выключение таймера

extern bool contGame;

clock\_t t0;

int baseTime, seconds;

void new\_game(){ //отрисовка массива и движок игры

settextstyle(SANS\_SERIF\_FONT, HORIZ\_DIR, 2);

if (!contGame){

CON\_TILES = 144;

baseTime = 0;

seconds = 0;

init\_game();

}

draw\_pole();

core\_game();

}

void maj\_init() //предварительное создание поля и его заполнение

{

for(int k = 0; k < he; k++)

for(int j = 0; j < wi; j++)

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

{

Pole[i][j][k].id = -1; //если значение id равно -1 -- эта позиция недоступна

switch(k){ //расставление мест где фишка должна быть

case 4:

if((i == 4 && j == 3) || (i == 4 && j == 2)) Pole[i][j][k].id = 0; //если id равен 0 -- позиция доступна

break;

case 3:

if ((i >= 3 && i <= 5) && (j >= 2 && j <= 4)) Pole[i][j][k].id = 0;

break;

case 2:

if ((i >= 2 && i <= 6) && (j >= 1 && j <= 5)) Pole[i][j][k].id = 0;

break;

case 1:

if ((i >= 2 && i <= 6)) Pole[i][j][k].id = 0;

else if ((i == 1) && (j >= 1 && j <= 5) || (i == 7) && (j >= 1 && j <= 5)) Pole[i][j][k].id = 0;

break;

case 0:

Pole[i][j][k].id = 0;

break;

}

}

//присвоение каждой структуре его координаты в массиве, координаты на поле и адрес для размещения картинки

for(int k = 0; k < he; k++)

for(int j = 0; j < wi; j++)

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

{

if (Pole[i][j][k].id == 0) {

Pole[i][j][k].id = layout[0].first;

strcpy(Pole[i][j][k].name, library[layout[0].first].name);

Pole[i][j][k].i = i;

Pole[i][j][k].j = j;

Pole[i][j][k].k = k;

Pole[i][j][k].x = begOfX + (i \* tileW) - k\*4 + 10;

Pole[i][j][k].y = begOfY + (tileH \* j) + k\*4;

Pole[i][j][k].bmp = library[layout[0].first].bmp;

layout.erase(layout.begin());

}

}

acc\_avl(); //пересчет доступных пар фишек

}

void draw\_pole(){ //отрисовывает фишки на поле, а также сколько осталось и сколько пар доступно

char output[50];

setVSPage();

clearviewport();

setcolor(WHITE);

sprintf(output, "Осталось фишек: %d", CON\_TILES); // вывод оставшихся фишек

outtextxy(350, 30, output);

sprintf(output, "Осталось ходов: %d", pairAVL); // вывод оставшихся ходов

outtextxy(560, 30, output);

sprintf(output, "Время: %02d:%02d", (seconds) / 60, (seconds) % 60); // вывод секундомера изменении окна

outtextxy(210 , 30, output);

for(int k = 0; k < he; k++) // выведение картинок фишек

for(int j = 0; j < wi; j++)

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

{

if (Pole[i][j][k].id != -1) {

putimage(Pole[i][j][k].x, Pole[i][j][k].y, Pole[i][j][k].bmp, TRANSPARENT\_PUT);}

}

putimage(10, 10, gMenu.bmp);

putimage(700, 300, findTiles.bmp);

setACPage();

}

void init\_game(){ // инициализация библиотеки и раскладки

setbkcolor(AVOCADO);

clearviewport();

lose.bmp = loadBMP("./MENU\_STUFF/lose.bmp");

win.bmp = loadBMP("./MENU\_STUFF/win.bmp");

gMenu.bmp = loadBMP("./MENU\_STUFF/gMenu.bmp");

findTiles.bmp = loadBMP("./MENU\_STUFF/find.bmp");

for (int i = 0; i < 42; i++) { // создание библиотеки фишек

library[i].id = i;

if (i < 34) library[i].count = 4;

else library[i].count = 1;

sprintf(library[i].name, "./TILES/tile%d.bmp", i+1);

library[i].bmp = loadBMP(library[i].name);

}

for (int i = 0; i < 42; i++) { for (int j = 1; j <= library[i].count; j++) { layout.push\_back(make\_pair(library[i].id, j)); } } //создание раскладки

shuffle(layout.begin(), layout.end(), rng); //реализация рандомизации(перемешивание раскладки)

maj\_init();

}

void core\_game() // основной процесс игры

{

t0 = clock();

int i1, i2, j1, j2, k1, k2; //для нахождения позиции в массиве

while(1)

{

if (pairAVL == 0) {end(); if(pereB) {pereB = false; break; }}

else{

if(definition\_XY(&i1, &j1, &k1)) break;

if(definition\_XY(&i2, &j2, &k2)) break;

delay(300);

if (i1 == i2 && j1 == j2 && k1 == k2) { draw\_pole(); continue; } //если одна и та же фишка то игнорируем

//если фишки одинаковы или они одинаковые как сезонные

if ((Pole[i1][j1][k1].id == Pole[i2][j2][k2].id || is\_season(Pole[i1][j1][k1].id, Pole[i2][j2][k2].id)) && is\_avalible(&Pole[i1][j1][k1]) && is\_avalible(&Pole[i2][j2][k2])){

delete\_pair(&Pole[i1][j1][k1], &Pole[i2][j2][k2]); // удаление фишек из массива

if (CON\_TILES != 0) acc\_avl(); //пересчет доступных пар фишек

}

}

if(CON\_TILES != 0) draw\_pole(); //отрисовка поля

else {victory(); break;}

}

}

bool definition\_XY(int \*i, int \*j, int \*k) // определение координат в массиве

{

(\*k) = -1;

if(click(i, j)) return true; // определение координат по XY и в массиве

for (int kn = he - 1; kn >= 0; kn--) { // определение на каком этаже находится фишка

if(Pole[\*i][\*j][kn].id == -1) continue;

else{(\*k) = kn; break;}

}

if (!is\_avalible(&Pole[\*i][\*j][\*k])) return false;

if (\*k != -1){setcolor(WHITE); border(&Pole[\*i][\*j][\*k]);}

return false;

}

void delete\_pair(TILE \*tile1, TILE \*tile2) // удаление фишек

{

TILE temp; temp.id = -1;

\*(tile1) = temp;

\*(tile2) = temp;

CON\_TILES -= 2;

}

bool is\_season(int tile1, int tile2) // проверка сезонная ли фишка

{

if(tile1 >= 34 && tile2 >= 34) return ((tile1 + 4) == tile2 || (tile2 + 4) == tile1);

return false;

}

bool is\_avalible(TILE\* tile1) //доступна ли фишка

{

int i = tile1->i, j = tile1->j, k = tile1->k;

if (((k == (he - 1)) || Pole[i][j][k+1].id == -1) && (( i == le - 1 || Pole[i + 1][j][k].id == -1) || (i == 0 || Pole[i - 1][j][k].id == -1))) return true;

return false;

}

bool click(int \*i, int \*j) // определение какую фишку выбрал пользователь

{

int x, y; // получение координат

do{

while(mousebuttons() != 1){ //подсчет и вывод секундомера

if (mousebuttons() != 1) stopwatch();

x = mousex();

y = mousey();

}

while(mousebuttons()==1); //поиск какая фишка в массиве

if (x >= 10 && x <= 100 && y >= 10 && y <= 55){

contGame = true;

seconds += baseTime % 60;

return true;

} // выход в меню

if (x >= 700 && x <= 750 && y >= 300 && y <= 350){find\_tiles();} // нахождение пар

}while(!(begOfX <= x && x <= begOfX + (tileW \* le)) || !(begOfY <= y && y <= begOfY + (tileH \* wi)));

if ((x < begOfX || x > begOfX + le\*tileW) && (y < begOfY || y > begOfY + wi\*tileH)) click(i, j);

\*i = ceil((x - begOfX) / tileW);

\*j = ceil((y - begOfY) / tileH);

return false;

}

void acc\_avl() //пересчет доступных пар фишек

{

vector<int> avl\_tile; //массив доступных фишек для их дальнейшего подсчета

int i = 0;

pairAVL = 0;

//занесение всех доступных фишек в массив

for(int k = 0; k < he; k++)

for(int j = 0; j < wi; j++)

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

if(Pole[i][j][k].id != -1 && is\_avalible(&Pole[i][j][k]))

{avl\_tile.push\_back(Pole[i][j][k].id); for\_find.push\_back(Pole[i][j][k]);}

sort(begin(avl\_tile), end(avl\_tile)); //сортировка по возрастанию

for(int i = 0; i < for\_find.size() - 1; i++) // нахождение 1 пары фишек и занесения в pair для функции find\_tiles()

for(int j = i + 1; j < for\_find.size(); j++)

if (for\_find[i].id == for\_find[j].id || is\_season(for\_find[i].id, for\_find[j].id)) {founds = make\_pair(for\_find[i], for\_find[j]); break;}

while((i + 1) < avl\_tile.size()) //подсчет доступных пар

{

if (avl\_tile[i] == avl\_tile[i + 1] || is\_season(avl\_tile[i], avl\_tile[i+1])){pairAVL++; avl\_tile.erase(avl\_tile.begin() + i, avl\_tile.begin() + i + 2);}

else i++; // здесь должна быть ошибка под конец игры

}

for\_find.clear();

}

void mix\_at\_end() // перемешивание при отсутсвующих фишках

{

TILE temp;

temp.id = 0;

vector <TILE> curTiles;

//зачистка массива

for(int k = 0; k < he; k++)

for(int j = 0; j < wi; j++)

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

if(Pole[i][j][k].id != -1) {curTiles.push\_back(Pole[i][j][k]); Pole[i][j][k].id = 0;}

shuffle(curTiles.begin(), curTiles.end(), rng); //новое перемешивание

//переприсвоение значений и координат

for(int k = 0; k < he; k++)

for(int j = 0; j < wi; j++)

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

if(Pole[i][j][k].id == 0) {

Pole[i][j][k] = curTiles[0];

Pole[i][j][k].i = i;

Pole[i][j][k].j = j;

Pole[i][j][k].k = k;

Pole[i][j][k].x = begOfX + (i \* tileW) - k\*4;

Pole[i][j][k].y = begOfY + (tileH \* j) + k\*4;

curTiles.erase(curTiles.begin());

}

acc\_avl();

draw\_pole();

}

void border(TILE \*tile) // границы при нажатии на фишку(не работает с swapbuffers())

{

rectangle(tile->x, tile->y, tile->x + tileW, tile->y + tileH);

}

void find\_tiles() // нахождение фишек если пользователь их не видит

{

setcolor(PROMPT);

border(&founds.first); border(&founds.second);

TILE temp; temp.id = -1;

temp.x = 800; temp.y = 600;

founds.first = temp;

founds.second = temp;

}

void printSW() // вывод секундомера

{

setcolor(WHITE);

char s[50];

sprintf(s, "Время: %02d:%02d", (baseTime + seconds) / 60, (baseTime + seconds) % 60);

outtextxy(210 , 30, s);

}

void stopwatch() // реализация секундомера

{

int dt = 0;

clock\_t t1 = clock();

baseTime = (int)((double)(t1 - t0) / CLOCKS\_PER\_SEC);

if (dt != baseTime){

printSW();

dt = baseTime;

}

}

void end() //окно при закончившихся доступных фишек

{

button but[3];

char s[25];

setVSPage();

clearviewport();

for(int i = 0; i < 3; i++) // указ координат кнопок и их адрес

{

sprintf(s, "./MENU\_STUFF/exit%d.bmp", i);

if(i != 0)

{

but[i].dx = 280; but[i].dy = 90;

but[i].x = 100 + but[i].dx\*(i - 1) + (i - 1)\*40; but[i].y = 450;

}

but[i].bmp = loadBMP(s);

}

for(int i = 0; i < 3; i++) // вывод экрана проигрыша и кнопок для выбора

{

if (i == 0) putimage(0, 0, but[i].bmp, COPY\_PUT);

else putimage(but[i].x, but[i].y, but[i].bmp, COPY\_PUT);

}

setACPage();

int flag = 0, x, y, st = 0; //какая кнопка была нажата

do {

while(mousebuttons() != 1){

x = mousex();

y = mousey();}

for(int i = 1; i < 3; i++){

if (x >= but[i].x && x <= but[i].x + but[i].dx && y >= but[i].y && y <= but[i].y + but[i].dy)

{flag = 1; st = i+1; break; }

}

}while(!flag);

if(st == 2) {mix\_at\_end(); pereB = false;}

else pereB = true;

}

void victory() // окно победы с выходом в главное меню

{

char res[100];

setVSPage();

clearviewport();

while(kbhit()) getch();

putimage(0, 0, win.bmp, COPY\_PUT); //выставление окна выигрыша

sprintf(res, "Ваше время прохождения: %d минут %d секунд!", (baseTime + seconds) / 60, (baseTime + seconds) % 60);

setcolor(BEIGE);

outtextxy(370, 300, res); // вывод времени прохождения

setACPage();

getch();

}