Міністерство освіти і науки України

Запорізька державна інженерна академія

Факультет автоматизованих систем управління виробництвом

Кафедра програмного забезпечення автоматизованих систем

**Контрольна робота**

**з дисципліни «Основи програмування»**

**на тему: «Обертальний квадрат»**

спеціальність 6.05010301 "Програмне забезпечення систем"

Виконав студент групи СП – 14 – 1д

Барабаш Андрій Андрійович

Керівник роботи:

доцент Скрипник І. А.

Запоріжжя

2015

ЗМІСТ

Вступ……………………………………………………………………………….....3

1 Опис………………………………………………………………………………...4

* 1. Функціональне призначення……………………………………………………4
  2. Опис логічної структури………………………………………………………..4
  3. Використовувані технічні і програмні засоби………………………………..10
  4. Функціональні вимоги…………………………………………………………10
  5. Вхідні та вихідна дані………………………………………………………….11
  6. Керівництво користувача………………………………………………………11

2Текст програми……………………………………………………………………12

3 Перелік посилань…………………………………………………………………25

**ВСТУП**

Дана програма була розроблена для засвоєння знань алгоритмів програмування та практичних навичок написання коду на мові С++, в тому числі використання таких елементів, як класи. Зміст програми – гра «Обертальний квадрат», де користувач повинен відсортувати літери в алфавітному порядку на деякому полі, за певними правилами перестановки. Також в програмі присутній алгоритм автоматичного сортування літер за правилами гри.

**1 ОПИС**

* 1. Функціональне призначення

Умова завдання:

Гра «Обертальний квадрат». Дано квадрат розміром 4 клітинки, в яких за допомогою датчика випадкових чисел розставлені літери від А до Р. Впорядкувати літери в квадраті за алфавітом. Квадрат має підквадрати, які можна обертати за годинниковою стрілкою на одну клітинку. Підквадрати мають розмір 2х2 і вказуються номером лівої верхньої клітинки. Існує операція, яка може бути виконана тільки один раз: обмін місцями двох літер.

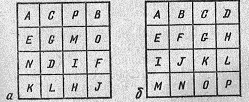


Рисунок 1.1. - гральні квадрати

У програмі надано можливість як самостійно впорядковувати клітинки, так й автоматично відсортувати літери за алфавітом за правилами гри з покроковою демонстрацією.

Постановка завдання

Необхідно було розробити програму, яка б реалізовувала гру за правилами (дозволяла обертати підквадрати, змінювати літери місцями), її інтерфейс, та можливість автогри (автоматичного сортування алфавіту).

Максимальний розмір квадрату - 4х4 клітинки, кількість літер – 16. Переміщення літер можливо лише за правилами гри.

* 1. Опис логічної структури

Ігрове поле представляє собою квадрат, який зручно розбити на 9 підквадратів (рис. 1.2). Кожен підквадрат за правилами гри ми можемо обертати за годинниковою стрілкою, тобто за один оберт кожна літера зміститься на 1 позицію. Таким чином, щоб перемістити літеру у верхню ліву комірку буде потрібно від 1 до 3 обертів, в залежності від початкового положення літери.

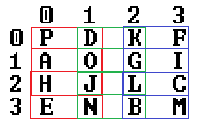


Рисунок 1.2 - Розбиття на підквадрати

Кожна комірка має свої координати – по рядкам и стовпчикам. Доцільно було б спочатку впорядкувати перші два рядки, кожного разу обертаючи підквадрат, в я кому знаходиться наша літера, переміщаючи її в верхню ліву комірку підквадрату. Для встановлення останніх літер кожного рядку потрібно щоб останні 2 літери були в одному підквадраті, який потім при обертанні встановлює зразу 2 літери.

Два останні рядки сортувати складніше, так як вони мають обмежену кількість підквадратів. В цьому випадку доречно використовувати можливість обміну двох літер, в тих випадках, коли сортування звичайними обертами не приводить до потрібного результату.

Алгоритм реалізує два способи використання програми – ручний й автоматичний. Програма складається з класу, полем якого є покажчик на двомірний масив – матрицю, яке є нашим основним квадратом. А також методів, які описують створення та видалення матриці (*Square()* конструктор та *~Square()* деструктор класу), вивід матриці та інтерфейсу користувача на екран (*Output()*), випадкове розставлення літер (*Random()*), ручну та автоматичну гру(*Game(), Autogame()*).

З самого початку користувач задає кількість випадкових переміщень літер в матриці (метод *Random()*).

При ручній грі користувач власноруч сортує літери за алфавітом, при натисненні на клавіші обертається певний підквадрат (рис. 1.3), а також може

змінити дві будь-які літери місцями (метод класу *Swap()*). Таким чином користувач може самостійно «зіграти в гру» та відсортувати квадрат.

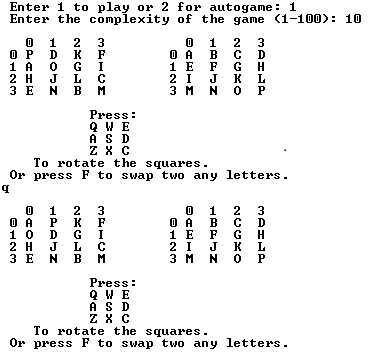


Рисунок 1.3 - Обертання підквадрата за натисненням клавіші

При виборі автоматичної гри (метод класу *Autogame()*) програма самостійно відсортовує та виводить результат кожного кроку алгоритму на

екран. В методі класу *Autogame()* в циклі проходимо нашу невідсортовану матрицю та знаходимо координати потрібної нам літери, починаючи з першої. Потім вибираємо в залежності від того, який рядок ми відсортовуємо, додаткові методи (*SortStr1(), SortStr2(), SortLastStr()*), за допомогою яких встановлюємо нашу літеру на потрібне місце.

Метод *SortStr1()* сортує перші s - 2 рядки плюс 2 літери передостаннього рядка, де s - розмір квадратної матриці. Для перших s – 2 літер кожного рядка, ми знаходимо координати верхньої лівої клітинки підквадрата (i, j), до якого належить наша літера, та викликаємо метод *Round(i,j)*, який обертає наш підквадрат по годинниковій стрілці на одну літеру. Таким чином, ми кожного разу виштовхуємо нашу літеру до гори (рис 1.4), та на кожній ітерації перевіряємо, чи знаходиться наша літера на потрібному місці.

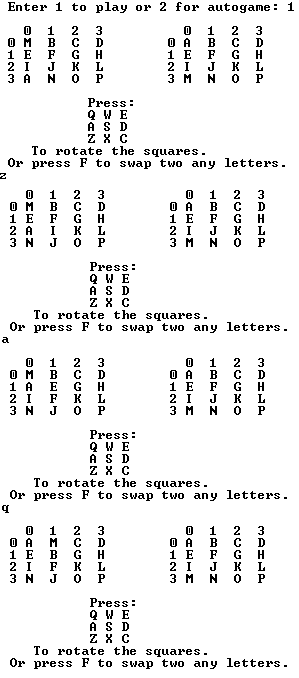


Рисунок 1.4 - Виштовхування літери на потрібне місце

Для останніх двох літер кожного з цих рядків ми використовуємо наступний алгоритм: передостанню літеру встановлюємо спочатку на останнє

місце в рядку, потім останню літеру встановлюємо прямо під нею, і обертаємо наш підквадрат, встановлюючи таким чином наші літери на потрібні нам місця (рис 1.5). Перші дві літери передостаннього рядку ми також обертаємо «на бік», це нам буде потрібно для сортування останнього рядку (рис 1.6).

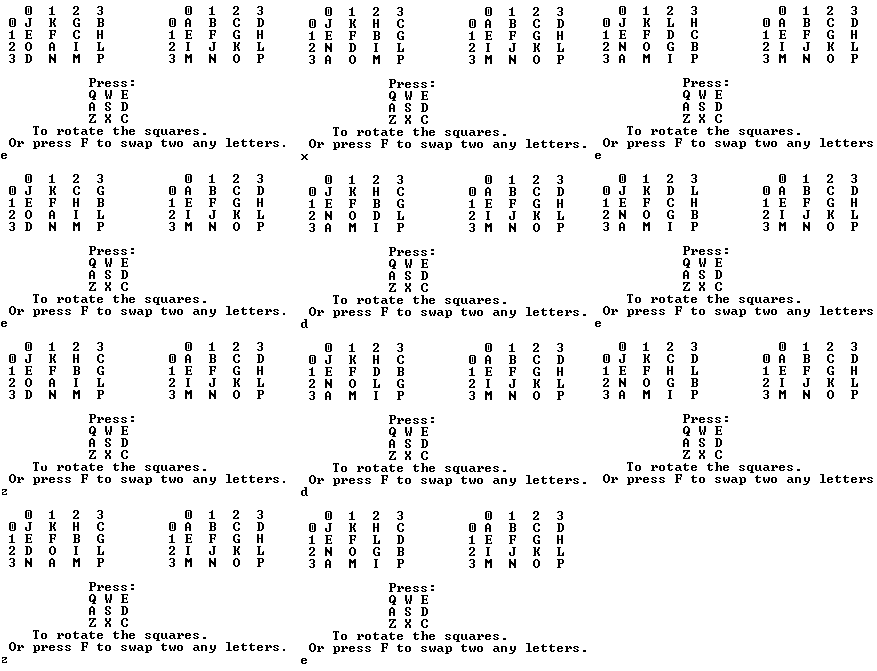


Рисунок 1.5 – Сортування останніх двох літер в рядку

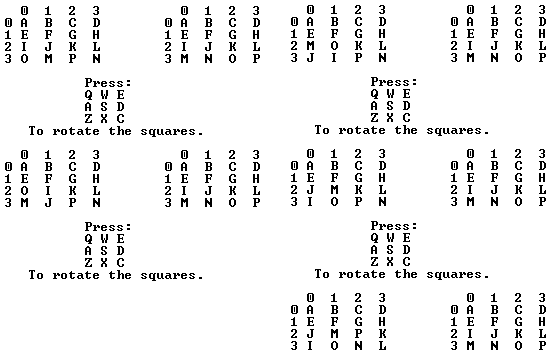


Рисунок 1.6 - Обертання літер "на бік"

Метод *SortStr2()* сортує останні 2 літери передостаннього рядку. Спочатку встановлюємо останню літеру на потрібне місце, потім передостанню на своє, і також обертаємо їх «на бік».

Останнім йде метод *SortLastStr()*. В нас залишився невідсортованим останній рядок нашої матриці, тим паче він у нас знаходиться у вигляді підквадрату, розташованого в центрі двох останніх рядків (рис. 1.7). На кожній ітерації методу ми перевіряємо, чи можливо обміняти 2 літери цього підквадрату (методом *Swap()*) так, щоб останній рядок бу у відсортованому вигляді, для цього хоча б дві літери з останнього рядку повинні бути на своєму місці, якщо це можливо – змінюємо літери місцями, та обертаємо підквадрати, повертаючи літери, які ми раніше встановлювали «на бік», на своє місце, разом з цим и

встановлюється останній рядок (рис. 1.8). Якщо ж літери останнього рядку не на своїх місцях, ми обертаємо даний підквадрат, та робимо перевірку знову.

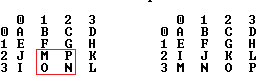


Рисунок 1.7 - Підквадрат в якому знаходиться останній рядок

Отже, на кожній ітерації циклу методу *Autogame()*  ми встановлюємо одну літеру на потрібне місце, й перевіряємо, чи відсортувався наш квадрат, за допомогою методу *Check()*.

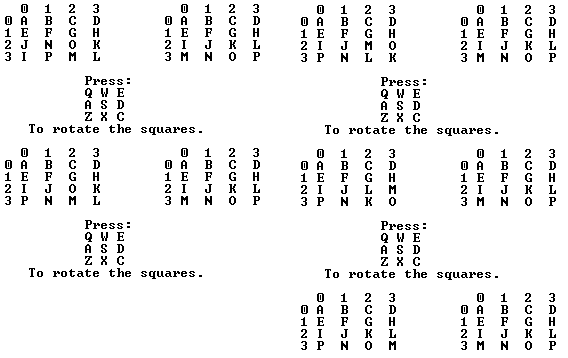


Рисунок 1.8 - Встановлюємо останній рядок

* 1. Використовувані технічні і програмні засоби

Для функціонування програми потрібен персональний комп’ютер з встановленою операційною системою Windows 7 і вище. Програма написана на мові високого рівня С++ за допомогою IDE Visual Studio 2013.

1.4 Функціональні вимоги

Програма складається з одного класу *Square* який містить десять методів: *Random(), Output(), Swap(), Check(), Round(), SortStr1(), SortStr2(), SortLastStr(), AutoGame(), Game().*

Сигнатури методів:

bool f – перевіряє, чи була вже виконана зміна літер місцями.

bool Win – перевіряє, чи відсортований масив.

bool end – перевіряє, чи літера на своєму місці.

bool pos – перевіряє, чи передостання літера в рядку на місці останньої

bool high – перевіряє, чи потрібно підвищити верхню границю.

int x, y – координати верхньої лівої комірки підквадрата.

int m, n – координати літери в відсортованому квадраті.

int i, j – координати літери в не відсортованому квадраті.

int border – ліва границя положення літери.

int hborder – верхня границя положення літери.

int count – кількість відсортованих літер на даний момент.

int a – код літери.

Діаграма класів представлена на рисунку 1.9.

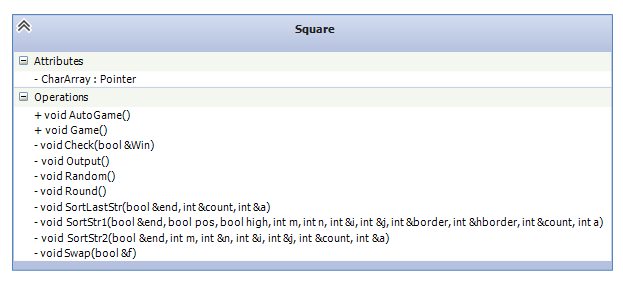


Рисунок 1.9 - Діаграма класів

1.5 Вхідні та вихідна дані

Вхідні дані представляють собою числові значення для вибору типу гри (1 або 2), для вибору кількості випадкових переставлень літер (1....100), а також символів q, w, e, a, s, d, z, x, c, f які користувач вводить для обертання певного підвкадрата.

Вихідні дані – дві квадратні матриці з координатною сіткою, одна з яких є робочою, в якій проводять переміщення літер, а друга – вже відсортованою, як приклад користувачеві. Також виводиться підказка у вигляді літер, які користувач повинен натискати для обертання під квадратів.

1.6 Керівництво користувача

На початку гри, користувач має вибрати її тип натиснувши відповідну цифру - 1 або 2. Одиниця для ручної гри ,двійка для автоматичної.

При виборі автоматичної гри, програма відсортує покроково літери в квадраті, та виведе результати на екран.

При виборі ручної гри, користувач може натискати певні клавіші, які відповідають координатам лівої верхньої комірки (q (0,0), w(0,1), e(0,2), a(1,0), s(1,1), d(1,2), z(2,0), x(2,1), c(2,2)) для обертання потрібних підквадратів, та клавішу f для обміну місцями двох літер. Коли квадрат буде відсортовано, на екрані з’явиться переможний напис, та запит на продовження програми.

2 ТЕКСТ ПРОГРАМИ

Вміст файлу Square.h

#ifndef SQUARE\_H

#define SQUARE\_H

#include <iostream>

#include <time.h>

using namespace std;

const int s = 4;//размер массива

typedef char\* p;

typedef char\*\* pp;

class Square

{

private:

pp CharArray;

void Output();

void Swap(bool& f);

void Check(bool& Win);

void Random();

void Round(int x, int y);

public:

Square();

~Square();

void AutoGame();

void Game();

void SortStr1(bool &end, bool pos, bool high, bool e, int m, int n, int &i, int &j, int &border, int &hborder, int &count, int a);

void SortStr2(bool &end, int m, int &n, int &i, int &j, int &count, int &a);

void SortLastStr(bool &end, int &count, int &a);

};

#endif

Вміст файлу Square.cpp

#include "Square.h"

Square::Square()

{

CharArray = new p[s];

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

{

CharArray[i] = new char[s];

}

char mas = 'A';

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

{

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

{

//cin >> mas;//откоментить для ввода алфавита вручную + закоментить тогда Random

//cin.get();

CharArray[i][j] = mas;

++mas;

}

}

}

Square::~Square()

{

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

{

delete[] CharArray[i];

CharArray[i] = nullptr;

}

delete[]CharArray;

CharArray = nullptr;

}

void Square::Random()

{

int z;

srand(time(NULL));

cout << " Enter the complexity of the game (1-100): ";

while (!(cin >> z)||z<0||z>100)

{

cin.clear();

while (cin.get() != '\n')

continue;

cout << "Error input! Reinput value:\n";

}

for (int k = 0; k < z; k++)//random

{

int i = (rand() % s);

int j = (rand() % s);

int m = (rand() % s);

int n = (rand() % s);

int c = CharArray[i][j];

CharArray[i][j] = CharArray[m][n];

CharArray[m][n] = c;

}

cin.get();

}

void Square::Output()

{

int mas = 'A';

for (int i = 0; i < s; i++)//для массива

{

cout << endl;

if (i == 0)//для сетки координат

{

cout << " ";

cout << " ";

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

{

cout << k << " ";

}

cout << " ";

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

{

cout << k << " ";

}

cout << endl;

}

cout << " ";

cout << i << " ";

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

{

cout <<CharArray[i][j] << " ";

}

cout << " ";

cout << i << " ";

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

{

cout << (char)mas << " ";

++mas;

}

}

cout << endl;

cout << endl;

cout << " Press: " << endl;

cout << " Q W E" << endl;

cout << " A S D" << endl;

cout << " Z X C" << endl;

cout << " To rotate the squares." << endl;

}

void Square::Swap(bool& f)

{

while (f)

{

char a, b;

cout << "Enter first upper letter: ";

cin.get(a);

if (cin.get() != '\n')

while (cin.get() != '\n'){};

if (a < 'A' || a>'P')

{

cout << "Wrong input!" << endl;

break;

}

cout << "Enter second upper letter: ";

cin.get(b);

if (cin.get() != '\n')

while (cin.get() != '\n');

if (a < 'A' || a>'P')

{

cout << "Wrong input!" << endl;

break;

}

int m, n, c;

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

{

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

{

if (CharArray[i][j] == a)

{

m = i;

n = j;

break;

}

}

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

{

if (CharArray[i][j] == b)

{

c = CharArray[m][n];

CharArray[m][n] = CharArray[i][j];

CharArray[i][j] = c;

}

}

}

f = false;

}

}

void Square::Check(bool& Win)

{

int mas = 'A';

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

{

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

{

if (CharArray[i][j] != mas)

{

Win = false;

break;

}

++mas;

}

}

if (Win)

cout << endl << " VICTORY!" << endl;

}

void Square::Round(int x, int y)

{

char c = CharArray[x][y];

CharArray[x][y] = CharArray[x + 1][y];

CharArray[x + 1][y] = CharArray[x + 1][y + 1];

CharArray[x + 1][y + 1] = CharArray[x][y + 1];

CharArray[x][y + 1] = c;

Output();

}

void Square::SortStr1(bool& end, bool pos, bool high, bool e, int m, int n, int& i, int& j, int& border, int& hborder, int& count, int a)

{

end = (m == i&&n == j); pos = false;

if (border == (s - 2) && (i == m&&n == (j - 1)))//если предпоследний элемент С на нужной позиции

{

pos = true; end = true; ++hborder; high = true;

}

if ((hborder == (s - 2)) && (i < (s - 1)) && (a + 1 == CharArray[i + 1][j]) && end && (count >= ((s - 2)\*s)))//I J если J под I

{

Round(i, j); Round(i, j); Round(i, j); Round(i, j + 1); Round(i, j); Round(i, j + 1);

}

if (end)

{

if (border == (s - 2) && !pos&&CharArray[i][j + 1] != a + 1)//если С не на нужной позиции и след за ним буква - следущая в цикле

{

Round(i, j); ++hborder; high = true; pos = true; j++;//сортируем предпослений элемент С

}

else

if (border == (s - 2) && !high)//повышаем верхнюю границу

++hborder;

if (CharArray[i][j - 1] == (a + 1) && pos) //проверяем не являеться ли A B D C

{

Round(i, j - 1); Round(i, j - 1); Round(i, j - 1); Round(i + 1, j - 1); Round(i, j - 1); Round(i + 1, j - 1); //крутим что б убрать D

}

}

while (j < border && !end)//пока не выпехнем элемент за границу

{

if (i == (s - 1))

{

Round(i - 1, j); Round(i - 1, j);

if (j == 0 && border == s - 2)//если G на координатах s-1 0 частный случай

{

Round(i - 1, j + 1); e = true;

}

--i; ++j;

}

else

{

Round(i, j); ++j; if (e){ ++j; }

}

}

if (!end)

if (border == (s - 1))//ставим крайний элемент в нужное место D

{

if (i == hborder&&j == (s - 1))//если сразу на месте, проварачиваем подквадрат

{

--i; --j; Round(i, j); Round(i, j); Round(i, j); end = true;

}

else

{

while (i != hborder || j != (s - 1))

{

--i; --j; Round(i, j); Round(i, j); Round(i, j); ++j;

}

--i; --j; Round(i, j); Round(i, j); Round(i, j); end = true;

}

}

if (!end)

i--; j--;//находим координаты вершины

if (!end)

if (j < border)//выталкиваем за границу

{

j = border; Round(i, j);

}

else

if (i < hborder)

{

i = hborder; Round(i, j); Round(i, j); Round(i, j);

}

else

{

Round(i, j); Round(i, j);

}

if (count == (s - 2)\*s + 2 && end)//крутим I J на край (бок)

{

Round(i, j); Round(i, j); Round(i, j);

}

}

void Square::SortStr2(bool &end, int m, int &n, int &i, int &j, int &count, int &a)

{

while (count > ((s - 2)\*s + (s - 2)) && count < ((s - 1)\*s + 1))//если сортируем 2 последних элемента предпоследней строки

{

if (count == ((s - 2)\*s + s - 1))//меняем значения дефолтных координат тк сначала сортируем L потом K

++n;

if (count == ((s - 2)\*s + s))//меняем значения дефолтных координат тк сначала сортируем L потом K

{

--n; --n;

}

end = (m == i&&n == j);//проверка на координаты на своем ли месте буква

if (end)

{

break;

}

if (count == ((s - 2)\*s + s) && i == s - 1 && j == s - 1)//если К под L исключение

{

--i; --j; Round(i, j);

--j; Round(i, j); Round(i, j);

++j; Round(i, j); Round(i, j); Round(i, j);

--j; Round(i, j);

end = true;

break;

}

if (i == s - 1)//Проверяем в какой строке нижней или предпоследней

{

if (j >= s - 2)//проверяем за серединой или нет

{

--i; --j; Round(i, j); Round(i, j); Round(i, j); ++j;

end = (m == i&&n == j);

if (end)

break;

else

Round(i, j); j++;

}

else

{

--i; Round(i, j); Round(i, j); ++j;

end = (m == i&&n == j);

if (end)

break;

else

Round(i, j); j++;

}

}

else

{

Round(i, j); ++j;

end = (m == i&&n == j);

if (end)

break;

else

Round(i, j); ++j;

}

end = (m == i&&n == j);

if (end)

{

break;

}

}

if (count == ((s - 2)\*s + s - 1))//возвращаемся к К

{

--a; --a;

}

if (count == ((s - 2)\*s + s))//поварачиваем K L - крайние элементы на бок

{

++a; Round(s - 2, s - 2);

}

}

void Square::SortLastStr(bool &end, int &count, int &a)

{

while (count >((s - 2)\*s + s))//если осталась последняя строка

{

int check = 0; int x = -1, y = 0;//координаты

if (CharArray[s / 2 + 1][s / 2 - 1] == a)//проверяем, на своем ли месте элементы в последней строке

{

++check;

}

if (CharArray[s / 2][s / 2 - 1] == a + 1)

{

++check;

}

if (CharArray[s / 2][s / 2] == a + 2)

{

++check;

}

if (CharArray[s / 2 + 1][s / 2] == a + 3)

{

++check;

}

if (check == s)//если все элементы на месте

{

end = true; Round(s - 2, 0); Round(s - 2, s / 2); Round(s - 2, s / 2); Round(s - 2, s / 2); a = a + s; break;

}

if (check < 2)//если меньше 2 на месте то прокручиваем

{

Round(s - 2, s / 2 - 1);

}

else

{

Round(s - 2, 0); Round(s - 2, s / 2); Round(s - 2, s / 2); Round(s - 2, s / 2); //если хотя бы 2 на месте - возвращаем их в строку

if (CharArray[s - 1][0] != a)

x = 0;

if (CharArray[s - 1][1] != a + 1)//проверям какие 2 буквы нужно сменить местами

if (x != -1)

y = 1;

else

x = 1;

if (CharArray[s - 1][2] != a + 2)

if (x != -1)

y = 2;

else

x = 2;

if (CharArray[s - 1][3] != a + 3)

if (x != -1)

y = 3;

int w = CharArray[s - 1][x];//меняем эти 2 буквы

CharArray[s - 1][x] = CharArray[s - 1][y];

CharArray[s - 1][y] = w;

a = a + s;//что б основной цикл дальше не шел

cout << " SWAP\n";

Output();

end = true;//конец цикла

break;

}

}

}

void Square::AutoGame()

{

Random();//сортируем рандомно наш массив

Output();

bool pos = false; bool end = false; int m = 0, n = -1; int border = -1; int hborder = 0; int count = 0; bool e = false; bool high = false;

for (int a = 'A'; a < 'P'; a++)//a - current character

{

++n;//для сравнения с дэфолтным алфавитом вместо еще одного массива просто задаем координаты текущей буквы

if (border < (s - 1))//левая граница нужна для выталкивания за нее буквы

++border;

else

{

border = 0;//обнуляем границу когда начинается новая строка

++m;//следующая строка

n = 0;//с начала строки

}

count++;//количество отсортированных букв

end = false; high = false; e = false;//обнуляем булевcкие переменные

if (count == ((s - 2)\*s + s - 1))//если дошли до К сортируем сначала L

++a;

int i = 0;

while (i < s && !end)

{

int j = 0;

while (j < s&&!end)

{

while (CharArray[i][j] == a && !end)//если наша буква является текущей

{

if (count < s\*(s - 2) + (s - 1))//сортируем s-2 строки + 2 элемента s-1 строки

{

SortStr1(end, pos, high, e, m, n, i, j, border, hborder, count, a);

}

else

if (count < s\*(s - 1) + 1)//сортируем последние 2 элемента s-1 строки

SortStr2(end, m, n, i, j, count, a);

else//сортируем последнюю строку

SortLastStr(end, count, a);

}

++j;

}

++i;

}

}

Check(end);

}

void Square::Game()

{

bool Win = false, f = true; char k;

Random();

Output();//for output matrix

while (!Win)

{

if (f)

cout << " Or press F to swap two any letters." << endl;

cin.get(k);//Здесь можно сделать, что б вводить координаты вместо буквы

if (cin.get() != '\n')

while (cin.get() != '\n');

switch (k)

{

case('q') :

Round(0, 0);//здесь можно передавать координаты, которые ввели выше

break;

case('w') :

Round(0, 1);

break;

case('e') :

Round(0, 2);

break;

case('a') :

Round(1, 0);

break;

case('s') :

Round(1, 1);

break;

case('d') :

Round(1, 2);

break;

case('z') :

Round(2, 0);

break;

case('x') :

Round(2, 1);

break;

case('c') :

Round(2, 2);

break;

case('f') :

if (f)

Swap(f);//for swap 2 letters

else

cout << "Alredy used!" << endl;

break;

default:

cout << "Reinput" << endl;

}

Win = true;

Check(Win);//check the Victory

}

}

Вміст файлу Main.cpp

#include "Square.h"

const int NotUsed = system("color F0");

void main()

{

char y = 'y'; int i; Square x;

while (y == 'y')

{

cout << " Enter 1 to play or 2 for autogame: ";

while (!(cin >> i) || i < 1 || i>2)

{

cin.clear();

while (cin.get() != '\n')

continue;

cout << "Error input! Reinput value:\n";

}

switch (i)

{

case 1:

x.Game();

break;

case 2:

x.AutoGame();

break;

}

cout << "\n Again? y/other" << endl;

cin >> y;

}

}

3 ПЕРЕЛІК ПОСИЛАНЬ

1. Прата Стивен «Язык программирования С++. Лекции и упражнения, 6-е изд.», Пер. с англ – М.:ООО «И.Д. Вильямс», 2012.-1248 с. :ил. – Парал. тит.англ

2. Джесс Либерти «С++ за 21 день. Третье издание» 815 с.

3. Бондарев В.М. «Программирование на С++. 2-е изд.» Харьков: «Компания СМИТ», 2005. – 284 с.