Лабораторная работа №7

Студента групп ПИ-202

Сычева Павла Николаевича

Выполнение:\_\_\_\_\_\_\_\_ Защита:\_\_\_\_\_\_\_\_

**ОБРАБОТКА ДИНАМИЧЕСКИХ СПИСКОВ ДАННЫХ**

**Цель работы:** ознакомиться с организацией многомерных динамических массивов в языке С/С++; приобрести практические навыки в применении односвязных линейных списков при обработке данных из внешних файлов.

**Содержание работы**

**Для части А:**

Разработать составной тип данных, для хранения информации согласно заданию. Разработать функции для инициализации переменных составного типа данных.

**Для части Б:**

Разработать динамическую структуру данных согласно заданию на основе структуры из части А. Реализовать с помощью динамической структуры задачу согласно заданию. В программе структуры данных не должны храниться в динамических массивах. Ввод исходных данных осуществлять из файла. Результаты работы программы выводить на экран

**Ход работы**

**Вариант №14**

**Для части А:**

**Задание:** Информация об одном ходе шахматной партии: тип фигуры, цвет фигуры, начальная позиция, конечная позиция.

**Для части Б:**

**Задание:** Структура данных «односвязный список - очередь». В списке ходов одной шахматной партии определить конечную позицию первой фигуры, которая находится в списке

**Часть А**

**Блок-схема**



Основная блок-схема

Функция read\_move

Входные параметры: turn

Выходные параметры: нет

Назначение: Ввод данных о ходе



Функция read\_move

Функция print\_move

Входные параметры: turn

Выходные параметры: нет

Назначение: Выводит информацию о ходе



Функция print\_move

1.Написал код программы:

#include <iostream>

#include <fstream>

#include <math.h>

#include <windows.h> ///Нужен для смены кодировок и работы кириллицы

#define N 100

using namespace std;

struct mv{

char type[40], clr[20];

char sPos[3], ePos[3];

};

void read\_move(mv& turn){

cin >>turn.type;

OemToCharA(turn.type, turn.type);

cin >>turn.clr;

OemToCharA(turn.clr, turn.clr);

cin >>turn.sPos;

OemToCharA(turn.sPos, turn.sPos);

cin >>turn.ePos;

OemToCharA(turn.ePos, turn.ePos);

}

void print\_move(mv& turn){

cout <<"Фигура -- " <<turn.type <<'\n';

cout <<"Цвет -- " <<turn.clr <<'\n';

cout <<"Начальная позиция -- " <<turn.sPos <<'\n';

cout <<"Конечная позиция -- " <<turn.ePos <<'\n';

}

int main(){

setlocale(LC\_ALL, "Russian");

cout <<"Введите данные о ходе:\n(фигура, цвет, начальная и конечные позиции)\n";

mv turn;

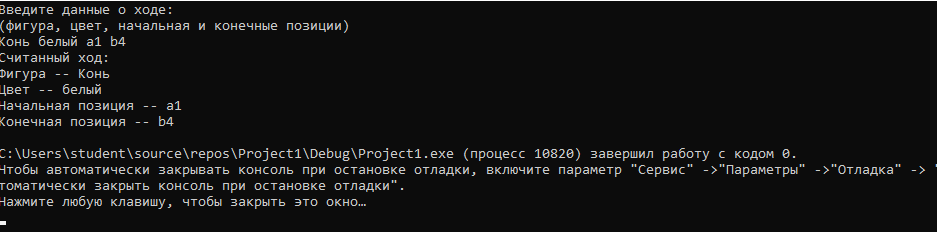
read\_move(turn);

cout <<"Считанный ход:\n";

print\_move(turn);

}

2.Запустил и протестировал программу:



Результат работы программы

**Часть 2**

**Блок-схема**



Основная блок-схема

Функция read\_move

Входные параметры: turn, in

Выходные параметры: нет

Назначение: считывает с файла информацию



Функция read\_move

Функция new\_node

Входные параметры: first\_elem, elem, nullptr

Выходные параметры: res

Назначение: создаёт новый узел



Функция new\_node

Функция create\_list

Входные параметры: turn

Выходные параметры: res

Назначение: создаёт новый список



Функция create\_list

Функция add

Входные параметры: the\_list, turn

Выходные параметры: нет

Назначение: Добавляет информацию о ходе в список



Функция add

Функция del

Входные параметры: нет

Выходные параметры: нет

Назначение: удаляет информацию о первым о ходе



Функция del

Функция wander

Входные параметры: the\_list

Выходные параметры: нет

Назначение: производит обход списка



Функция wander

1.Написал код программы:

#include <iostream>

#include <fstream>

#include <math.h>

#include <windows.h> ///Нужен для смены кодировок и работы кириллицы

#define N 100

#define nullptr 0

using namespace std;

struct mv {

char type[40], clr[20];

char sPos[3], ePos[3];

};

void read\_move(mv& turn, ifstream& in) {

in >> turn.type;

OemToCharA(turn.type, turn.type);

in >> turn.clr;

OemToCharA(turn.clr, turn.clr);

in >> turn.sPos;

OemToCharA(turn.sPos, turn.sPos);

in >> turn.ePos;

OemToCharA(turn.ePos, turn.ePos);

}

void print\_move(mv& turn) {

cout << "Фигура -- " << turn.type << '\n';

cout << "Цвет -- " << turn.clr << '\n';

cout << "Начальная позиция -- " << turn.sPos << '\n';

cout << "Конечная позиция -- " << turn.ePos << '\n';

}

struct node {

mv val;

node\* next;

};

node\* new\_node(mv val, node\* next) {

node\* res = new node;

res->val = val;

res->next = next;

return res;

}

struct list {

node\* first;

node\* last;

int amount;

};

list create\_list(mv first\_elem) {

list res;

res.first = new\_node(first\_elem, nullptr);

res.last = res.first;

res.amount = 1;

return res;

}

void add(list& lst, mv elem) {

if (lst.amount != 0) {

node\* new\_nd = new\_node(elem, nullptr);

lst.last->next = new\_nd;

lst.last = new\_nd;

lst.amount++;

}

else {

cout << "...Ошибка\n";

//...

}

}

void del(list& lst) {

if (lst.amount < 2) {

node\* new\_first = lst.first->next;

delete lst.first;

lst.first = new\_first;

lst.amount--;

}

}

void wander(list& lst) {

node\* current = lst.first;

char\* last\_end\_pos = current->val.ePos;

current = current->next;

bool was\_eaten = false;

while ((current != nullptr) && (!was\_eaten)) {

if (strcmp(last\_end\_pos, current->val.sPos) == 0)

last\_end\_pos = current->val.ePos;

else if (strcmp(last\_end\_pos, current->val.ePos) == 0)

was\_eaten = true;

current = current->next;

}

cout << "Конечная позиция первой фигуры:\n";

cout << last\_end\_pos;

}

int main() {

setlocale(LC\_ALL, "Russian");

mv turn;

ifstream in("in.txt");

read\_move(turn, in);

list the\_list = create\_list(turn);

while (!in.eof()) {

read\_move(turn, in);

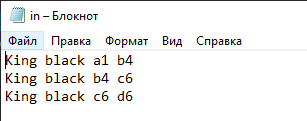
add(the\_list, turn);

}

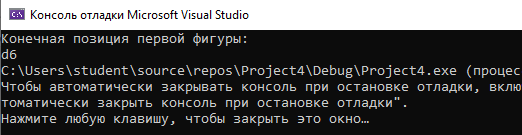
wander(the\_list);

}

2.Запустил и протестировал программу:



Тестовые данные



Результат работы

Вывод: В языке С ++ хорошо реализована работа со списками, что упрощает работы с большим количеством данных.