**МИНОБРНАУКИ РОССИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»**

**(НГТУ)**

Институт радиоэлектроники и информационных технологий Кафедра «Прикладная математика»

Лабораторная работа №1

**Выполнил:**

Студент 1-го курса, 19-ПМ-2

Дубков Иван

**Проверил:**

Доцент кафедры «Прикладная математика»

Чернов Антон Григорьевич

Нижний Новгород 2020

## Содержание

1. Цель………………………………………………………………………………………3
2. Описание классов………………………………………………………………………..4
3. Скриншоты работы программы………………………………………………………...5
4. Исходный код…………………..………………………………………………………..6
   1. Mylib.h………………………………………………………………………….6
   2. Funclaba1.cpp………………………………………………………………..…7
   3. Main.cpp……………………………………………………………………....10
5. Вывод…………………………………………………………………………………...11

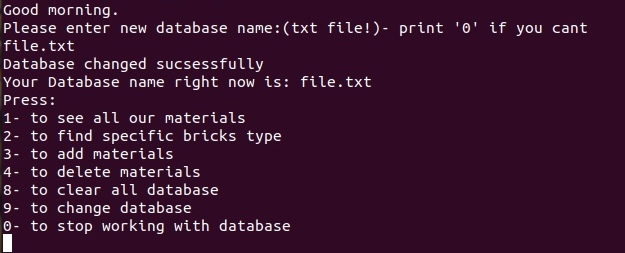
# Цель

Создать программу для автоматизации работы кирпичного склада. В программе каждый вид кирпичей должен иметь определенные параметры, такие как форма, цвет и количество кирпичей этого вида на складе. Она должна давать возможность получать информацию о различных параметрах кирпичей: выполнять поиск по характеристикам и т.д. Кроме того, программа должна давать пользователю удобный и понятный интерфейс.

# Возможности

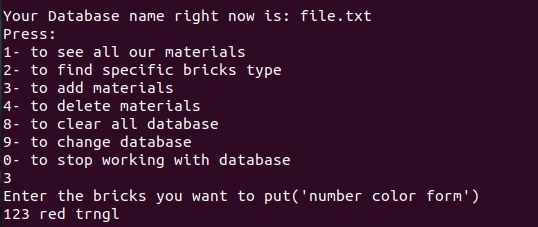
* Хранение и редактирование базы кирпичного склада.
* Выполнение поиска по кирпичам (по цвету и форме).
* Добавление кирпичей, а также их удаление.
* Отображение информации о текущем наполнении склада.
* Удаление всей информации.

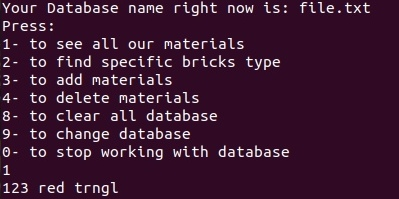
# Работа программы

Меню:

:

:

Добавление кирпичей:

Просмотр содержимого склада:

# Исходный код

Mylib.h

#include <cstdlib>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Bricks{

public:

int num;

string form;

string colour;

friend std::ostream& operator<< (std::ostream &out, const Bricks &br);

friend std::istream& operator>> (std::istream &in, Bricks &br);

};

std::ostream& operator<< (std::ostream &out, const Bricks &br);

void add (string namefile);

void search(string namefile);

void deleting(string namefile);

void allwrite(string namefile);

void allclear(string namefile);

Funclaba1.cpp

#include "mylib.h"

using namespace std;

std::ostream& operator<< (std::ostream &out, const Bricks &br)

{

out << br.num << " " << br.colour << " " << br.form<< endl;

return out;

}

std::istream& operator>> (std::istream &in, Bricks &br)

{

in >> br.num;

in >> br.colour;

in >> br.form;

return in;

}

void add (string namefile)

{//Добавление объектов на склад(либо новый тип, либо плюс к старому)

ifstream fin;

ofstream fout;

fin.open(namefile);

int i=0;

int k=0;

string s;

while (!fin.eof()){//Считаем количество строк в тексте=количество видов объектов

getline(fin, s);

k++;

}

k--;

Bricks \*a = new Bricks[k+1];

fin.close();

fin.open(namefile);

for (i=0; i<k; i++){

fin >> a[i];//Считываем элементы из файла, заодно выводим на экран все текущие элементы

}

int num, j;

int tmp=-1;//Флаг для поиска совпадений

string colour;

string form;

cin >> num >>colour>> form;//Непосредственно, сколько каких кирпичей хотят добавить

for (i=0; i<k; i++){

if ((a[i].form== form)&&(a[i].colour== colour)){//Сравниваем параметры

a[i].num=a[i].num+num;

tmp=i;//Меняем флаг

}

}

fin.close();//Готовимся полностью очистить файл

if (tmp==-1){//Флаг не поменялся-> Не было совпадений -> Дописываем в конец

a[k].num= num;//заполняем предусмотрительно выделенную лишнюю ячейку в массиве введенными данными

a[k].colour=colour;

a[k].form=form;

fout.open(namefile, ios::out);//Мы снова в файле, но он теперь пустой

for (i=0; i<k+1; i++){//Переписываем весь дополненный массив в документ

if ((i<k)&&(a[i].num>0)){

fout << a[i];

}

else {

fout<< a[k];

}

}

}

else {//Флаг поменялся->Было совпадение ->Переписываем сам массив

fout.open(namefile, ios::out); //Открываем пустой файл

for (i=0; i<k; i++){//Переписываем в файл массив, нужная строчка уже исправлена

fout << a[i];

}

}

fout.close();//Убираем за собой

delete [] a;//Убираем за собойх2

}

void deleting(string namefile){//Удаление элементов

ifstream fin;

ofstream fout;

fin.open(namefile);

int i=0;

int k=0;

string s;

while (!fin.eof()){

getline(fin, s);

k++;

}

k--;

Bricks \*a = new Bricks[k+1];

fin.close();

fin.open(namefile);

for (i=0; i<k; i++){

fin >> a[i];//Считываем элементы из файла, заодно выводим на экран все текущие элементы

}

int num;

int tmp=-1;

string colour;

string form;

cin >> num >>colour >> form;//Считываем данные об удаляемом типе объектов

for (i=0; i<k; i++){

if ((a[i].form== form)&&(a[i].colour== colour)){//Ищем соответствующие признаки

if (num>a[i].num){//Если просят удалить больше, чем есть на складе

tmp=-2;

}

if (a[i].num>=num) {//Удаление кирпичей через уменьшение параметра количества на введенное значение

a[i].num=a[i].num-num;

tmp=i;

}

}

}

if (tmp>-1){//Если мы уменьшали количество

fin.close();

fout.open(namefile, ios::out);//открываем пустой файл

for (i=0; i<k; i++){//Запись измененного массива в файл

if(a[i].num>0){//Доп. условие - если количество равно нулю, записано не будет

fout << a[i];

}

}

}

fout.close();

delete[]a;

}

void search(string namefile){//Удаление элементов

ifstream fin;

ofstream fout;

fin.open(namefile);

int i=0;

int k=0;

string s;

while (!fin.eof()){

getline(fin, s);

k++;

}

k--;

Bricks \*a = new Bricks[k+1];

fin.close();

fin.open(namefile);

for (i=0; i<k; i++){

fin >> a[i];//Считываем элементы из файла, заодно выводим на экран все текущие элементы

}

int num;

int tmp=-1;

string colour;

string form;

cin >>colour >> form;//Считываем данные об удаляемом типе объектов

for (i=0; i<k; i++){

if ((a[i].form== form)&&(a[i].colour== colour)){//Ищем соответствующие признаки

cout << a[i];

tmp=1;

}

}

if (tmp==-1){

}

fout.close();

delete[]a;

}

void allwrite(string namefile){//Вывести все элементы на экран

ifstream fin;

fin.open(namefile);

string s;

while (!fin.eof()){//Считаем количество строк в тексте=количество видов объектов

getline(fin, s);

cout<<s<<endl;

}

fin.close();//Убираем за собой

}

void allclear(string namefile){//Очистить файл с данными

ofstream f;

f.open(namefile, ios::out);//EZ

f.close();

}

Main.cpp

#include "mylib.h"//Подключаем заголовочный файл

using namespace std;

int main(int argc, char \*argv[]){

string namefile;

namefile=argv[1];

cout << "Good morning."<<endl;

string s;//На случай смены файла с данными

int n;

int k=0;

fstream file1; fstream file2;

while (k==0){//Цикл по k

cout <<"Your Database name right now is: "<< namefile<<endl;

cout <<"Press:"<<endl;

cout <<"1- to see all our materials"<<endl;

cout <<"2- to find specific bricks type"<<endl;

cout <<"3- to add materials"<<endl;

cout <<"4- to delete materials"<<endl;

cout <<"8- to clear all database"<<endl;

cout <<"9- to change database"<<endl;

cout <<"0- to stop working with database"<<endl;

cin >> n;

void (\*func)(string namefile);//Указатель на функцию

switch (n){

case 0:

exit(0);

break;

case 1:

func=allwrite;

break;

case 2:

func=search;

break;

case 3:

func=add;

break;

case 4:

func=deleting;

break;

case 8:

func=allclear;

break;

case 9:

cout <<"Please enter new database name:(txt file!)"<<endl;

cin >> s;

file2.open(s);

if(!file2){

cout <<"Sorry, database can not be found. Please try again"<<endl;

}

else{

cout<<"Database changed sucsessfully"<<endl;

namefile=s;

}

file2.close();break;

default:

cout<<"You uncorrect data. Please, try again"<<endl;

break;

}

if ((n==1)||(n==2)||(n==3)||(n==4)||(n==8)){

func(namefile); //Вызов соответствующей функции

}

}

return 0;

}

# Вывод

В результате лабораторной работы была получена программа для автоматизации работы склада кирпичей. Поставленные условия были выполнены.