**ЛАБОРАТОРНАЯ РАБОТА №1**

**«ИССЛЕДОВАНИЕ ОСОБЕННОСТЕЙ ОБЪЕКТНОГО ПОДХОДА ПРИ ПРОГРАММИРОВАНИИ СЛОЖНЫХ СТРУКТУР ДАННЫХ С ДИНАМИЧЕСКИМИ ПОЛЯМИ»**

**Цель работы**: Исследование основных средств определения класса, создания объектов класса, приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей использования конструкторов копирования.

**Вариант задания**

Разработать программу на языке C++. Работа программы должна быть реализована в виде меню со следующими пунктами:

- создание объекта;

- создание копии объекта (используя конструктор копирования);

- просмотр полей оригинала и копии;

- изменение одного из полей объекта (копии или оригинала);

- выполнение задания согласно варианту.

Вариант 11

Динамическая структура — двусвязный список. Хранимые данные — поставки железной руды на плавильную печь: номер поставки (число), вес руды (число) и ожидаемый выход металла (число 0.0-0.9). Предусмотреть функции добавления элементов в список и удаления из него, а также функцию поиска суммарного веса чистого металла.

**2. Код программы на языке C++**

#pragma warning(disable : 4996) ;

#pragma warning(disable : 6031) ;

#include <stdio.h>

#include <string.h>

#include <conio.h>

#include <windows.h>

#include <stdlib.h>

#define clear() system("cls")

struct postavka{

int id;

float mass;

float productivyty;

} ;

struct listPostavok {

postavka info;

struct listPostavok\* next;

struct listPostavok\* prev;

} ;

postavka\* InInfo();

class Postavki

{

public:

Postavki(Postavki & obj1);

Postavki();

~Postavki();

int AddNew(const postavka\* info);

int DellElem(const int id);

float GetMetalOutput();

int PrintInfo();

int CorrectInfo();

int GetCount() { return count; }

private:

int count;

listPostavok\* List;

listPostavok\* serchElemByID(const int id);

};

Postavki::Postavki()

{

List = NULL;

count = 0;

}

Postavki::~Postavki()

{

while (List->prev) List = List->prev;

while (List->next) {

List->next;

free(List->prev);

}

free(List);

}

int Postavki::AddNew(const postavka\* info) {

listPostavok\* pointer = NULL; if (!info) return 0;

if (!(List)) //если списка не существует - создать его

{

List = (listPostavok\*)calloc(sizeof(listPostavok), 1); //освобождаем память

pointer = List; //запоминаем

}

else

{

for (pointer = List; pointer->next; pointer = pointer->next); //идем до последнего

pointer = ((pointer->next = (listPostavok\*)calloc(sizeof(listPostavok), 1))->prev = pointer)->next; //создаем зависимость ссылок для элемента, сначала выделяем память и запоминаем, потом новому элементу прошлое задаем как текущее положение и потом переходим на этот поинтер.

}

pointer->info = \*info; //записываем данные

return 1;

}

listPostavok\* Postavki::serchElemByID(const int id) {

listPostavok\* elem = NULL;

if (List)

{

for (listPostavok\* tmp = List; tmp; tmp = tmp->next)

if (tmp->info.id == id)

{

elem = tmp;

break;

}

}

return elem;

}

int Postavki::DellElem(const int id) {

listPostavok\* elemToDel = NULL;

if (!(List)) return 0;

elemToDel = serchElemByID(id);

if (elemToDel)

{

if (elemToDel->next) elemToDel->next->prev = elemToDel->prev;

if (elemToDel->prev) elemToDel->prev->next = elemToDel->next;

if (List == elemToDel) List = elemToDel->next;

free(elemToDel);

return 1;

}

}

float Postavki::GetMetalOutput() {

if (!(List)) return 0;

listPostavok\* tmp = List;

float sum = 0;

while (tmp) {

sum += (tmp->info.mass) \* (tmp->info.productivyty);

tmp = tmp->next;

}

return sum;

}

int Postavki::PrintInfo() {

int Posid = 1;

if (!List) return 1;

while (List->prev) {

List = List->prev;

}

listPostavok\* tmp = List;

while (tmp != NULL) {

printf("Номер в списке :%d; ID:%d; Масса руды :%.2f ; Процент содержания металла: %.2f; \n",Posid,tmp->info.id, tmp->info.mass, tmp->info.productivyty);

tmp = tmp->next;

Posid++;

}

return Posid;

}

int Postavki::CorrectInfo() {

clear();

int SerchID; printf("\nВведите идентификационный номер -->"); scanf("%d", &SerchID);

int flag = 0; if (!List) return 0; postavka d, old;

while (List->prev) List = List->prev;

while (1)

{

d = List->info;

if (d.id == SerchID)

{

flag = 1;

old = d;

break;

}

if (List->next == NULL) break;

List = List->next;

}

if (!flag) {

printf("Не найдена запись с таким идентификатором .... Нажмите любую кнопку"); getch();

d.id = NULL;

return 0;

}

else {

printf("Найдена следующая запись :\n");

printf(" ID:%d; Масса руды :%.2f ; Процент содержания металла: %.2f; \n", List->info.id, List->info.mass, List->info.productivyty);

printf("Ввести новые данные ? --> (1/Any) "); int sel = 0; scanf("%d", &sel);

if (sel == 1) {

d = \*InInfo();

}

clear();

printf("Данные готовы к записи. Старая запись : \n");

printf(" ID:%d; Масса руды :%.2f ; Процент содержания металла: %.2f; \n", List->info.id, List->info.mass, List->info.productivyty);

printf("Новая запись \n");

printf(" ID:%d; Масса руды :%.2f ; Процент содержания металла: %.2f; \n", d.id, d.mass, d.productivyty);

printf("Сохранить новые данные ? --> (1/Any) "); scanf("%d", &sel);

if (sel == 1) {

List->info = d;

return 1;

}

else return 0;

}

}

Postavki::Postavki(Postavki &obj1) {

if (!obj1.List) return;

while (obj1.List->prev) obj1.List = obj1.List->prev;

if(List)

if(List->prev)

while (List->prev) List = List->prev;

if(List)

while (List->next) {

List->next;

free(List->prev);

}

free(List); count = 0;

while (obj1.List->next) {

AddNew(&obj1.List->info);

obj1.List = obj1.List->next;

count++;

}

AddNew(&obj1.List->info);

count++;

}

postavka\* InInfo() {

postavka temp;

printf("Введите индекс -->"); scanf("%d", &temp.id);

printf("\n Введите колличество руды -->");

scanf("%f", &temp.mass);

while (1) {

printf("\n Введите продуктивность(от 0.1 до 0.9) -->"); scanf("%f", &temp.productivyty);

if ((temp.productivyty >= 0.1) && (temp.productivyty <= 0.9)) break;

else {

printf("\n Введенно неверное значение продуктивности. Повтор ввода.");

}

}

return &temp;

}

int main()

{

SetConsoleCP(1251); // Задаем таблицу символов для консоли.

SetConsoleOutputCP(1251);

Postavki Postavka1; Postavki\* Postavka2 = new Postavki(); int id = 0;

while (1) {

clear();

printf("1 - Ввод записей в первый класс\n");

printf("2 - Удаление записи в первом классе\n");

printf("3 - Удаление записи в втором классе\n");

printf("4 - Редактировать данные первого класса \n");

printf("5 - Редактировать данные второго класса \n");

printf("6 - Вывести данные первого класса и второго класса\n");

printf("7 - Создать копию первого класса\n");

printf("8 - Вывести сумарное колличество метала из первого класса\n");

printf("9 - Вывести сумарное колличество метала из второго класса\n");

printf("\n");

printf("0 - Выход\n");

printf("Введите номер пункта -->"); int sel; scanf("%d", &sel); int temps = 0;

switch (sel)

{

case 1:

postavka temp;

while (1) {

printf("\nВведите номер поставки -->"); scanf("%d", &temp.id);

printf("\nВведите количество руды -->"); scanf("%f", &temp.mass);

while (1) {

printf("\nВведите продуктивность руды (значение от 0.1 до 0.9 -->"); scanf("%f", &temp.productivyty);

if ((temp.productivyty >= 0.1) && (temp.productivyty <= 0.9)) {

break;

}

else printf("\n Введено неверное значение. Допускается значения от 0.1 до 0.9. Введите верное значение");

}

if (Postavka1.AddNew(&temp)) printf("Запись успешно добавлена. Нажмите Esc для выхода из добавления ");

if (getch() == 27) break;

}

break;

case 2:

printf("\nВведите номер поставки, которую надо удалить -->"); scanf("%d", &id);

if (Postavka1.DellElem(id)) {

printf("\nЗапись успешно удалена ");

} else printf("\nТакая запись не существует либо возникла ошибка.... ");

getch();

break;

case 4:

Postavka1.CorrectInfo();

getch();

break;

case 5:

Postavka2->CorrectInfo();

getch();

break;

case 6:

clear();

printf("\nДанные с первого класса \n");

if( Postavka1.PrintInfo()== 1) printf("\nПервый класс пустой \n");

printf("\nДанные с Второго класса \n");

if (Postavka2->PrintInfo()== 1 ) printf("\nВторой класс пустой \n");

printf("\nНажмите любую кнопку\n");

getch();

break;

case 7:

Postavka2 = new Postavki(Postavka1);

break;

case 8:

printf("\n В первой поставке вышло следующее колличество металла --> %0.2f", Postavka1.GetMetalOutput());

getch();

break;

case 9:

printf("\n В Второй поставке вышло следующее колличество металла --> %0.2f", Postavka2->GetMetalOutput());

getch();

break;

case 3:

printf("\nВведите номер поставки, которую надо удалить с второй поставки-->"); scanf("%d", &id);

if (Postavka2->DellElem(id)) {

printf("\nЗапись успешно удалена ");

}

else printf("\nТакая запись не существует либо возникла ошибка.... ");

getch();

break;

break;

case 0:

exit(666);

break;

default:

break;

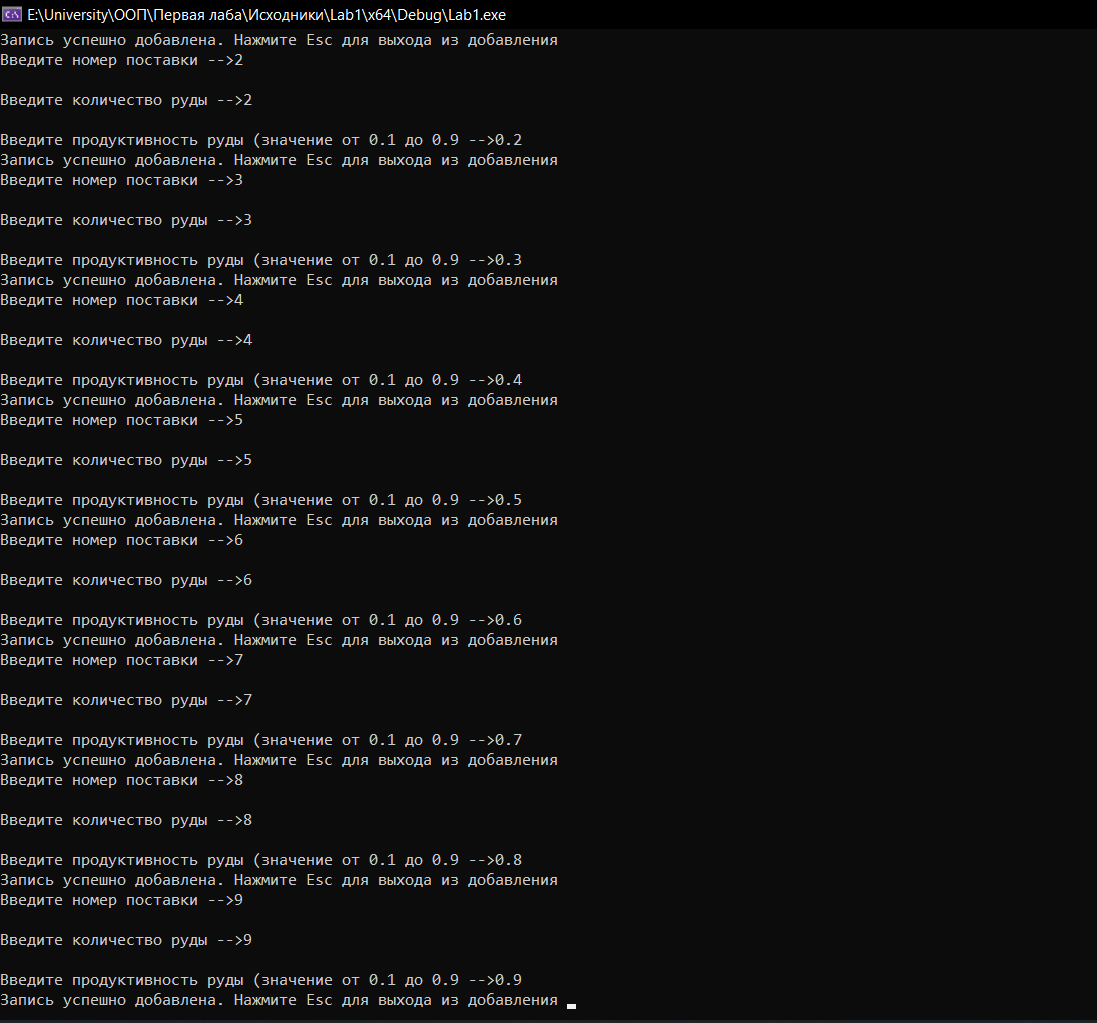
}

}

}

**3. Тестирование и отладка**

Тестирование программы будет проходить при использовании следующих данных. Данные будут числовые, каждая запись будет содержать данные исходя из id записи(Пример в Таблице 1).

Таблица 1 – Пример тестовых данных для проверки работоспособности программы.

|  |  |  |
| --- | --- | --- |
| Id | Количество руды | Продуктивность |
| 1 | 1 | 0.1 |
| 2 | 2 | 0.2 |
| 3 | 3 | 0.3 |
| 4 | 4 | 0.4 |
| 5 | 5 | 0.5 |
| 6 | 6 | 0.6 |
| 7 | 7 | 0.7 |
| 8 | 8 | 0.8 |
| 9 | 9 | 0.9 |

Рисунок 1 – Процесс ввода данных в первый класс

После ввода данных, выведем их на экран для проверки введенной информации. Поскольку вводили только в первый класс данные, то второй класс у нас на данный момент должен быть пустым.

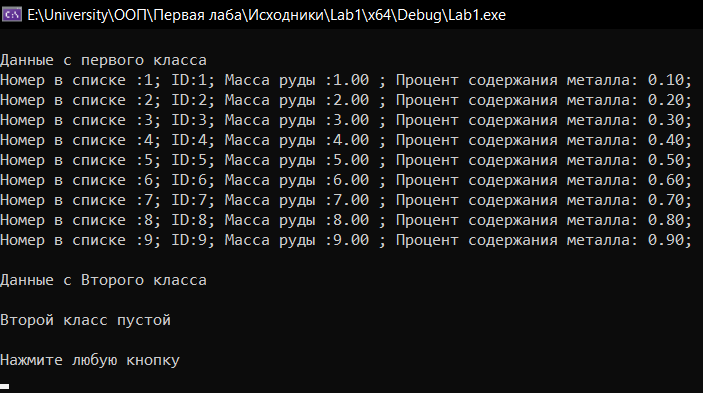


Рисунок 2 – Вывод информации на экран для проверки

Внесем изменения в данные первого класса, да бы проверить корректность работы методов класса.

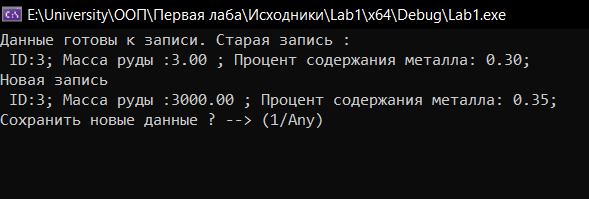


Рисунок 3 – Окно редактирования данных

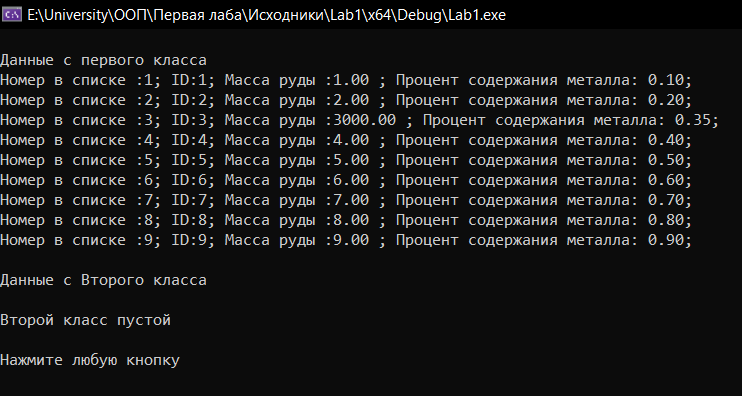


Рисунок 4 – Проверка корректности внесенных изменений

Далее по заданию, необходимо выполнить создание второго класса как копии первого. Вывести результат на экран, отредактировать данные второго класса, и показать что эти объекты – разные и работают корректно.

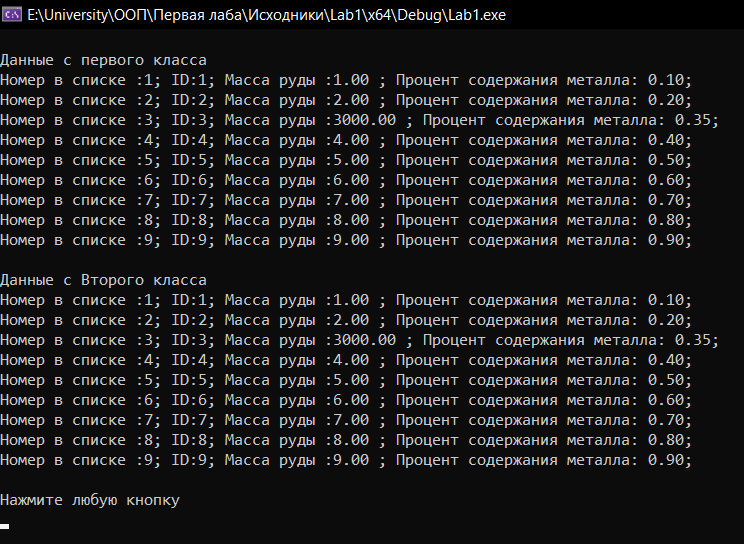


Рисунок 5 – Результат создания копии класса 1 в класс 2

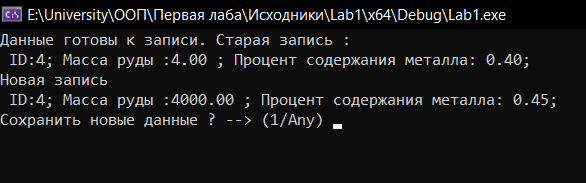


Рисунок 6 – Внесение изменений в запись во втором классе

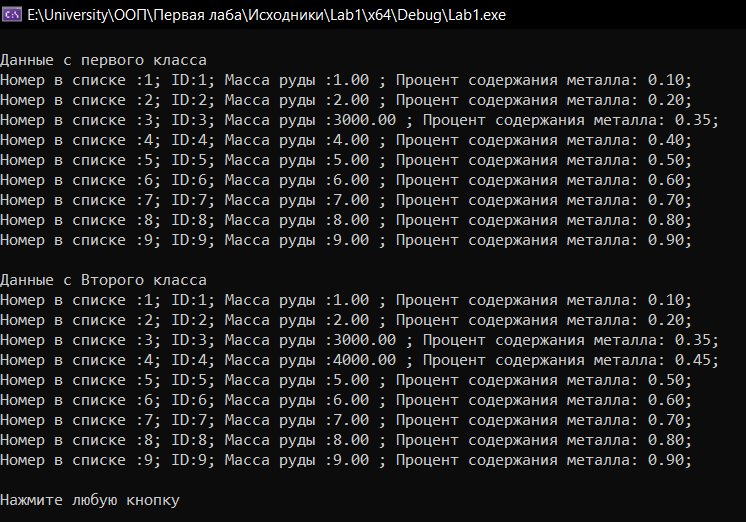


Рисунок 7 – Результат редактирования данных в классе 2

Далее выполним удаление по одной записи в первом и втором классе.

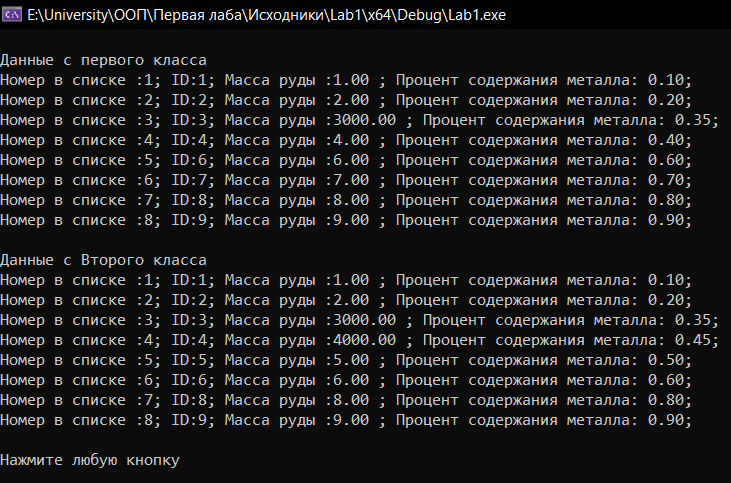


Рисунок 8 – Результат удаления записей из двух классов( из первого – запись с id=5, из второго с id=7)

Так же выполним проверку работоспособности подсчета суммарного выхода руды.

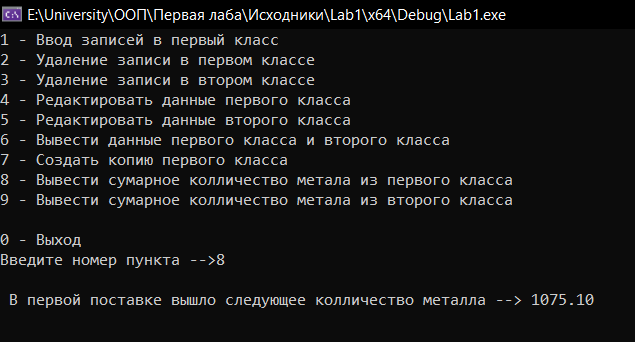


Рисунок 9 – Результат подсчета полного выхода руды из первого класса

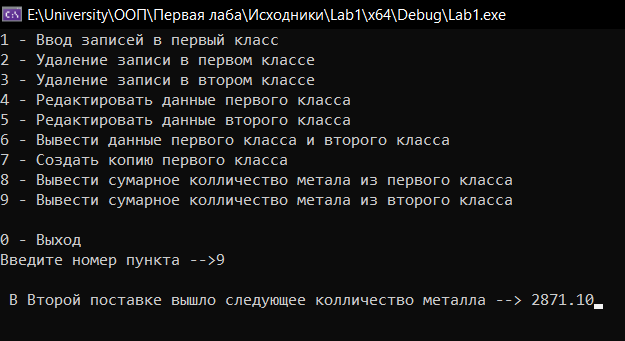


Рисунок 10 – Результат подсчета полного выхода руды из второго класса

В результате тестирования, видно, что классы работают правильно, копии классов создаются корректно, а значит конструкторы класса работают верно. Так же верно работают методы классов, необходимые для выполнения взаимодействия с данными, которые хранятся в них.

**Вывод**

При выполнении данной лабораторной работы были изучены методы работы с классами, были получены практические навыки по реализации класса, написанию конструкторов и деструкторов класса, включая конструктор копирования , написанию методов класса и выполнению работ с ними. Были приобретены навыки разработки и отладки программ, использующих динамическую память.