МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННО БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Учебный Центр Информационных Технологий «Информатика»

Лабораторная работа № 3

по дисциплине «Объектно-ориентированное программирование»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники и автоматизированных систем»

Выполнил слушатель: Бройтман Е.Д.
Вариант: №5
Дата сдачи:
Преподаватель: Силов Я.В.

Новосибирск, 2016г.

1. Задание

1) Разработать структуру абстрактного класса, который объявляет собой минимально необходимый интерфейс.
Минимальный интерфейс включает в себя функцию вывода.
2) Разработать производный класс, осуществив его наследование от разработанного абстрактного класса, с реализацией виртуальной функции.
Класс должен включить в себя:

- Поля - данные, для хранения значений.
- Реализацию виртуальной функции вывода.
- Static ДИНАМИЧЕСКИЙ массив указателей на объекты БАЗОВОГО (абстрактного) класса.
- Статическую переменную для хранения размера массива.
- Статические функции:
  - добавления указателя (типа Abstract****) на объект данного класса.
  - удаление элемента с заданным номером.
  - обход всего массива с вызовом функции вывода.

В пользовательском интерфейсе предоставить следующие возможности:

- создание нового объекта (и автоматическое добавление его в массив)
- ввод новых значений для объекта с заданным номером
- удаление по логическому номеру
- просмотр всего массива

ПРИМЕЧАНИЕ: для этих операций используются разработанные static методы.

2. Структурное описание

В данной лабораторной работе создаются базовый абстрактный класс **AbstractFraction** и наследуемый от него класс **Fraction**.
Для реализации методов классов и полей классов **AbstractFraction** и **Fraction** использован файл «Fraction.cpp».
В заголовочном файле «Fraction.h» создан интерфейс (тело) абстрактного класса AbstractFraction, интерфейс класса Fraction и приведен прототип функции void Menu().
В файле ”Menu.cpp” реализовано пользовательское меню, обеспечивающее:

- Создание нового объекта (и автоматическое добавление его в массив).
- Ввод новых значений для объекта с заданным номером.
- Удаление по логическому номеру.
- Просмотр всего массива.

В файле ”Main.cpp” реализованы вызовы функций обеспечения работы с кириллической кодировкой и функции меню.

Методы абстрактного класса **AbstractFraction** (все public):

- virtual void Print() = 0 - вывод на экран (чисто виртуальная функция).
- virtual ~AbstractFraction(){} - виртуальный деструктор для исключения утечки памяти при удалении производного объекта по указателю базового класса.
- virtual int & GetNumerator()=0 - извлечение значения числителя (чисто виртуальная функция).

- virtual int & GetDenominator()=0 - извлечение значения знаменателя (чисто виртуальная функция).

Поля и методы, реализованные в классе **Fraction**:
- int Numerator – числитель дроби (private).
- int Denominator - знаменатель дроби (private).
- static AbstractFraction **PFraction - static ДИНАМИЧЕСКИЙ массив указателей на объекты БАЗОВОГО (абстрактного) класса (private).
- static int Size - размер массива PFraction (private).
- Fraction() - конструктор по умолчанию (public).
- Fraction(const int N, const int D) - конструктор с параметрами (public).
- Fraction(const Fraction &F) - конструктор копирования (public).
- virtual ~Fraction() - виртуальный деструктор для исключения утечки памяти при удалении производного объекта по указателю базового класса.
- static void SetNumerator(const int N, const int Index) - установка значения числителя объекта, на который ссылается указатель массива PFraction по индексу Index.
- static void SetDenominator(const int D, const int Index) - установка значения знаменателя объекта, на который ссылается указатель массива PFraction по индексу Index.
- virtual int & GetNumerator() - извлечение ссылки на числитель (унаследованная чисто виртуальная функция).
- virtual int & GetDenominator() - извлечение ссылки на знаменатель (унаследованная чисто виртуальная функция).
- static int GetSize() - извлечение размера массива PFraction.
- virtual void Print() - вывод на экран (унаследованная чисто виртуальная функция).
- static void PrintAll() - вывод на экран всех объектов, указатели на которые находятся в PFraction.
- static void clearArray() - удаление всех объектов, на которые ссылаются указатели массива PFraction.
- static void delElem(const int n) - удаление объекта, на который ссылается указатель массива PFraction по индексу n.

3. Функциональное описание

Рассмотрим реализацию методов класса **Fraction**.
1. Подключаем наш заголовочный файл «Fraction.h».
2. Инициализируем статические поля:
- int Fraction::Size = 0;
- AbstractFraction ** Fraction::PFraction = NULL;

3. Конструктор класса по умолчанию **Fraction () –** запускается автоматически при создании нового объекта класса без параметров, инициализирует закрытые поля класса значением по умолчанию. Инициализация выполнена списком инициализации, значения по умолчанию **Numerator** = **1**, **Denominator** = **1**. Создается динамический массив указателей на абстрактный базовый класс размерностью на 1 больше, чем в массиве класса. Копирование всех указателей массива класса и указателя на новый объект в созданный массив. Статическая переменная **Size** увеличивается на **1**. Если массив класса не пустой – удаление массива класса. Копирование адреса созданного динамического массива в указатель на массив класса.

4. Конструктор с параметрами Fraction(const int N, const int D) - запускается автоматически при создании нового объекта класса с параметрами, инициализирует закрытые поля класса значениями, переданными в качестве параметров. Создается динамический массив указателей на абстрактный базовый класс размерностью на 1 больше, чем в массиве класса. Копирование всех указателей массива класса и указателя на новый объект в созданный массив. Статическая переменная **Size** увеличивается на **1**. Если массив класса не пустой – удаление массива класса. Копирование адреса созданного динамического массива в указатель на массив класса.
5. Конструктор копирования Fraction(const Fraction &F) - запускается автоматически при создании нового объекта класса как копии другого объекта, инициализирует закрытые поля класса значениями полей копируемого объекта. Создается динамический массив указателей на абстрактный базовый класс размерностью на 1 больше, чем в массиве класса. Копирование всех указателей массива класса и указателя на новый объект в созданный массив. Статическая переменная **Size** увеличивается на **1**. Если массив класса не пустой – удаление массива класса. Копирование адреса созданного динамического массива в указатель на массив класса.
6. virtual ~Fraction() - виртуальный деструктор запускается при удалении производного объекта класса Fraction по указателю на объект базового класса AbstractFraction.
Если в массиве указателей на объекты записан только один указатель:

- Удаляем массив с присвоением массиву значения NULL.
- Уменьшаем размер массива на 1.

Иначе, если количество указателей в массиве более 1 и удаляется элемент не с максимальным значением индекса в массиве:

- Создаем массив указателей на объекты базового класса AbstractFraction размерностью на 1 меньше размера массива-члена класса Fraction.
- Копируем в него указатели из массива-члена класса Fraction, за исключением указателя ссылающегося на удаляемый объект.
- Удаляем массив-член класса Fraction.
- Присваиваем указателю массива-члена класса Fraction указатель созданного массива указателей на объекты базового класса AbstractFraction.
- Уменьшаем размер массива на 1.

Иначе, если количество указателей в массиве более 1 и удаляется элемент с максимальным значением индекса в массиве:

- Уменьшаем размер массива на 1, таким образом удается избежать повторной перезаписи указателей из одного массива в другой при удалении каждого объекта, когда выполняется очистка массива.

7. static void SetNumerator (const int N, const int Index) - установка значения числителя объекта, на который ссылается указатель массива PFraction по индексу Index. С помощью реализованной виртуальной функции GetNumerator извлекаем ссылку на числитель объекта и копируем в числитель новое значение.
8. static void SetDenominator(const int D, const int Index) - установка значения знаменателя объекта, на который ссылается указатель массива PFraction по индексу Index. С помощью реализованной виртуальной функции GetDenominator извлекаем ссылку на знаменатель объекта и копируем в знаменатель новое значение.
9. virtual int & GetNumerator() - извлечение ссылки на числитель (унаследованная чисто виртуальная функция). Возвращает ссылку на числитель объекта, для которого вызывается метод.

10. virtual int & GetDenominator() - извлечение ссылки на знаменатель (унаследованная чисто виртуальная функция). Возвращает ссылку на знаменатель объекта класса **Fraction**, для которого вызывается метод.
11. static int GetSize() - извлечение размера массива PFraction. Возвращает значение статического поля класса **Fraction** содержащего размер массива указателей типа **AbstractFraction**** в классе **Fraction**.
12. virtual void Print() - вывод на экран (унаследованная чисто виртуальная функция). Выводит на экран числителя и знаменателя объекта класса **Fraction**, для которого вызывается метод.
13. static void PrintAll() - вывод на экран размера массива указателей типа **AbstractFraction**** в классе **Fraction** и всех объектов, указатели на которые находятся в этом массиве с помощью виртуальной функции Print.
14. static void clearArray() - удаление всех объектов, на которые ссылаются указатели массива указателей типа **AbstractFraction**** в классе **Fraction**, начиная с последнего элемента, заканчивая нулевым. При этом удается избежать повторной перезаписи указателей из одного массива в другой при удалении каждого объекта и вызове деструктора класса **Fraction**, в котором специально для этого случая предусмотрено только уменьшение на 1 содержимого переменной размера массива при размере >1 и удалении с конца.
15. static void delElem(const int n) - удаление объекта, на который ссылается указатель массива указателей типа **AbstractFraction**** в классе **Fraction** по индексу. Поскольку имеется виртуальный деструктор у базового и производного классов операция delete удаляет объект типа **Fraction**, при этом вызывается реализация деструктора для класса **Fraction**.

4. Описание работы программы

После запуска программы на экране появляется меню, показанное на Рис. 1.

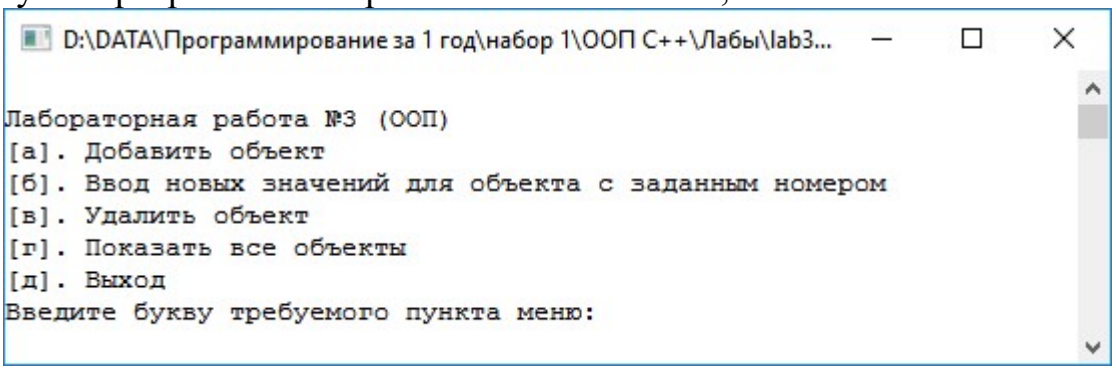

Рис. 1 Главное меню программы.

При выборе пункта а на экран будет выведено меню второго уровня добавления объектов. Рис. 2.

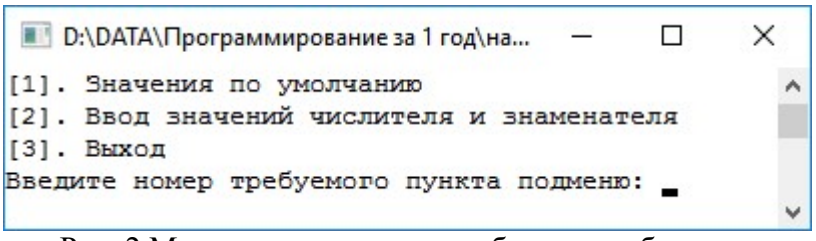

Рис. 2 Меню второго уровня добавления объектов.

При выборе пункта 1 меню второго уровня добавления объектов в массив будет помещен указатель на вновь созданный объект-дроби с числителем и знаменателем равными 1 и на экран будет опять выведено меню второго уровня добавления объектов.
При выборе пункта 2 меню второго уровня добавления объектов в массив будет предложено ввести значения числителя и знаменателя нового объекта-дроби. Рис. 3.

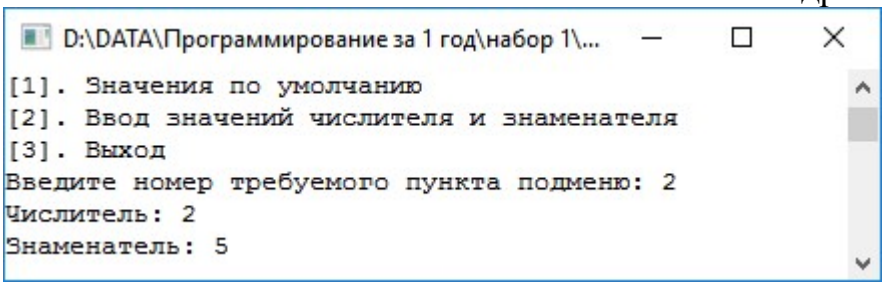

Рис. 3 Результат выбора пункта 2 меню второго уровня добавления объектов.

После ввода значений числителя и знаменателя нового объекта-дроби в массив будет помещен указатель на вновь созданный объект-дроби с введенными значениями числителя и знаменателя и на экран будет опять выведено меню второго уровня добавления объектов.
При выборе пункта 3 меню второго уровня добавления объектов на экране опять появится главное меню программы . Рис. 4.

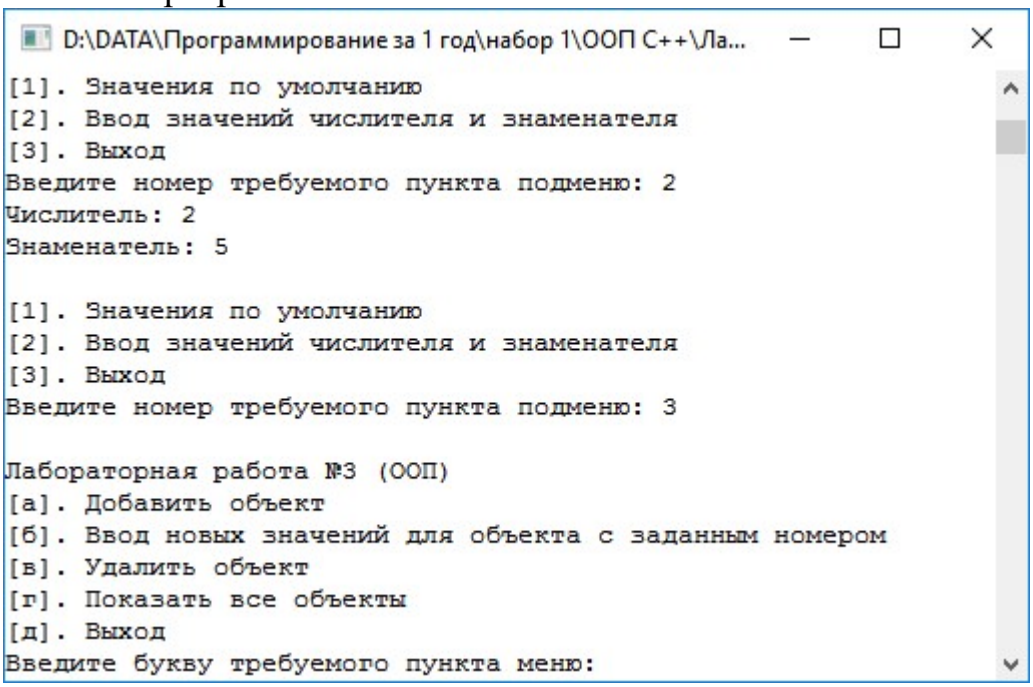

Рис. 4 Результат ввода значений числителя и знаменателя и выбора пункта 3 меню второго уровня добавления объектов.

При выборе пункта г главного меню программы на экран будут выведены количество записей в массиве, значения объектов-дробей, указатели на которые хранятся в массиве, и главное меню программы. Рис. 5.

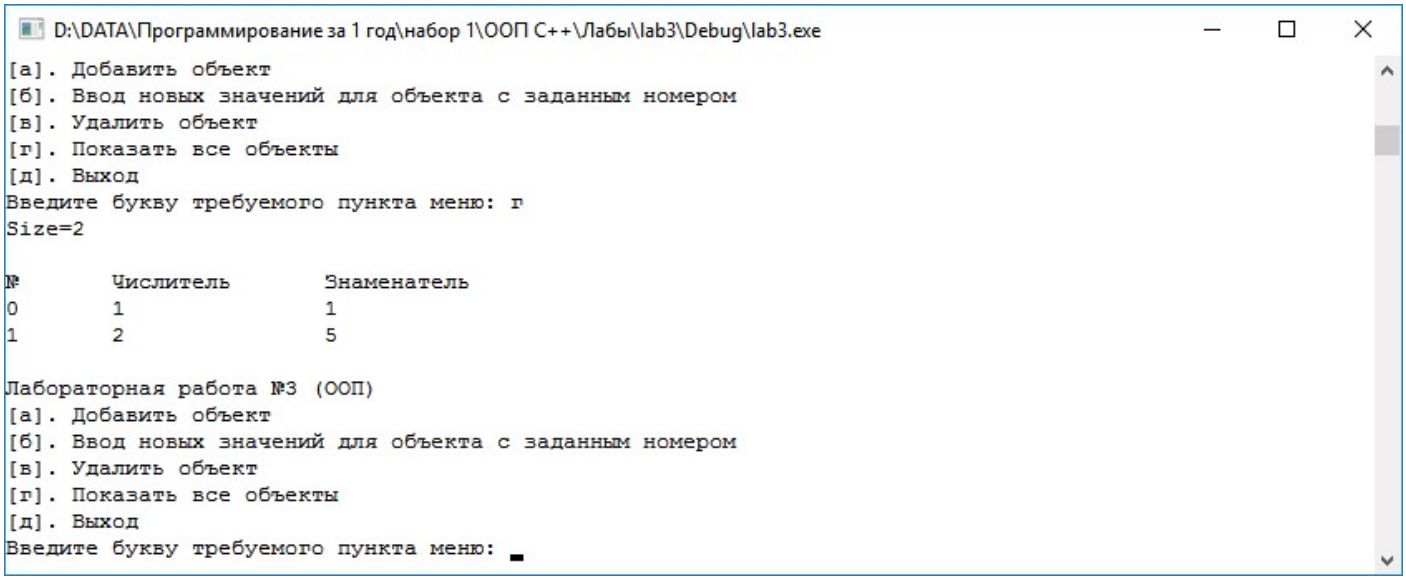

Рис. 5 Результат выбора пункта г главного меню программы.

При выборе пункта б главного меню программы будет предложено ввести номер объекта и новые значения числителя и знаменателя этого объекта-дроби. Рис. 6.

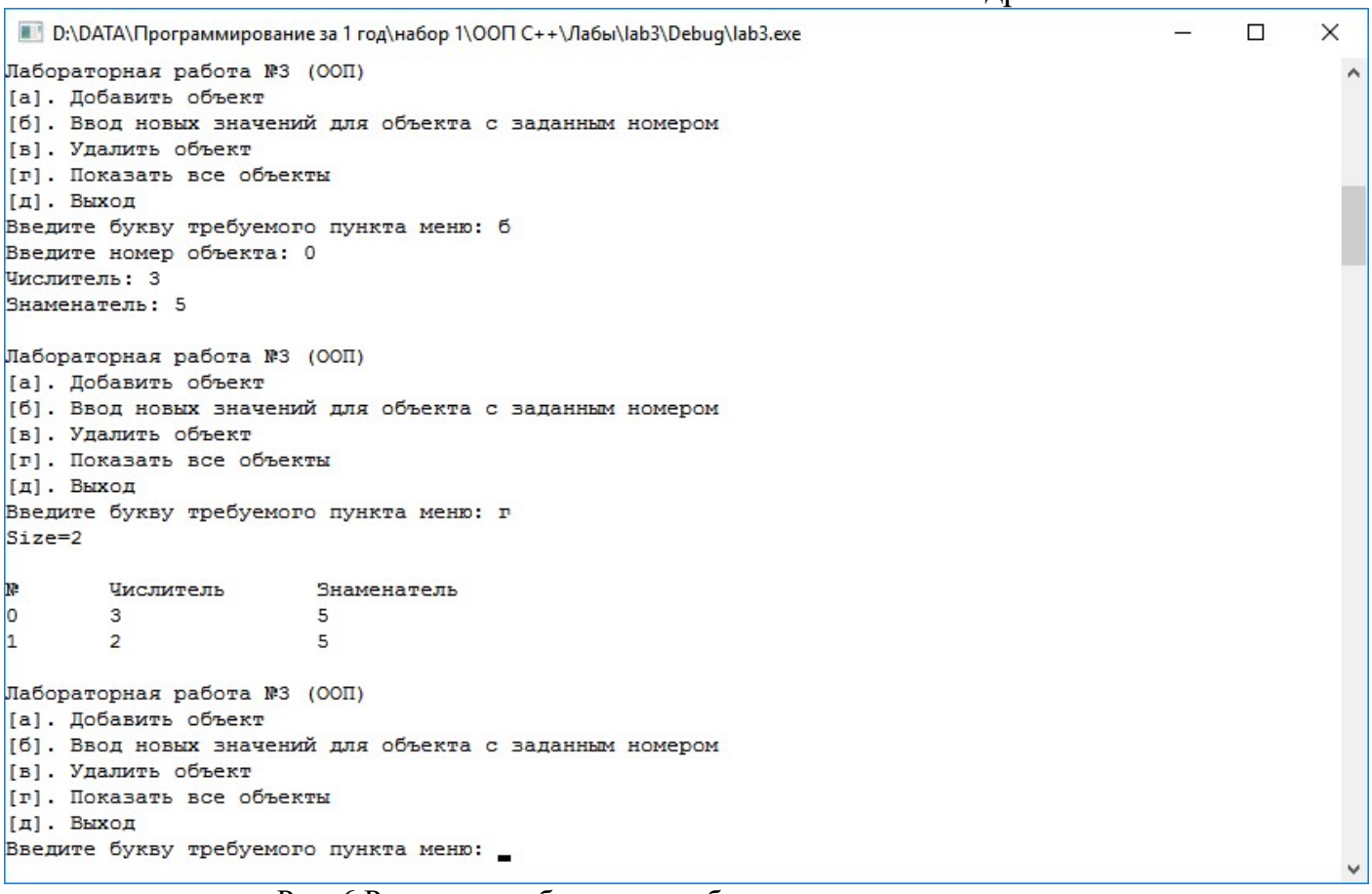

Рис. 6 Результат выбора пункта б главного меню программы.

При выборе пункта в главного меню программы будет предложено ввести номер удаляемого объекта. Рис. 7.

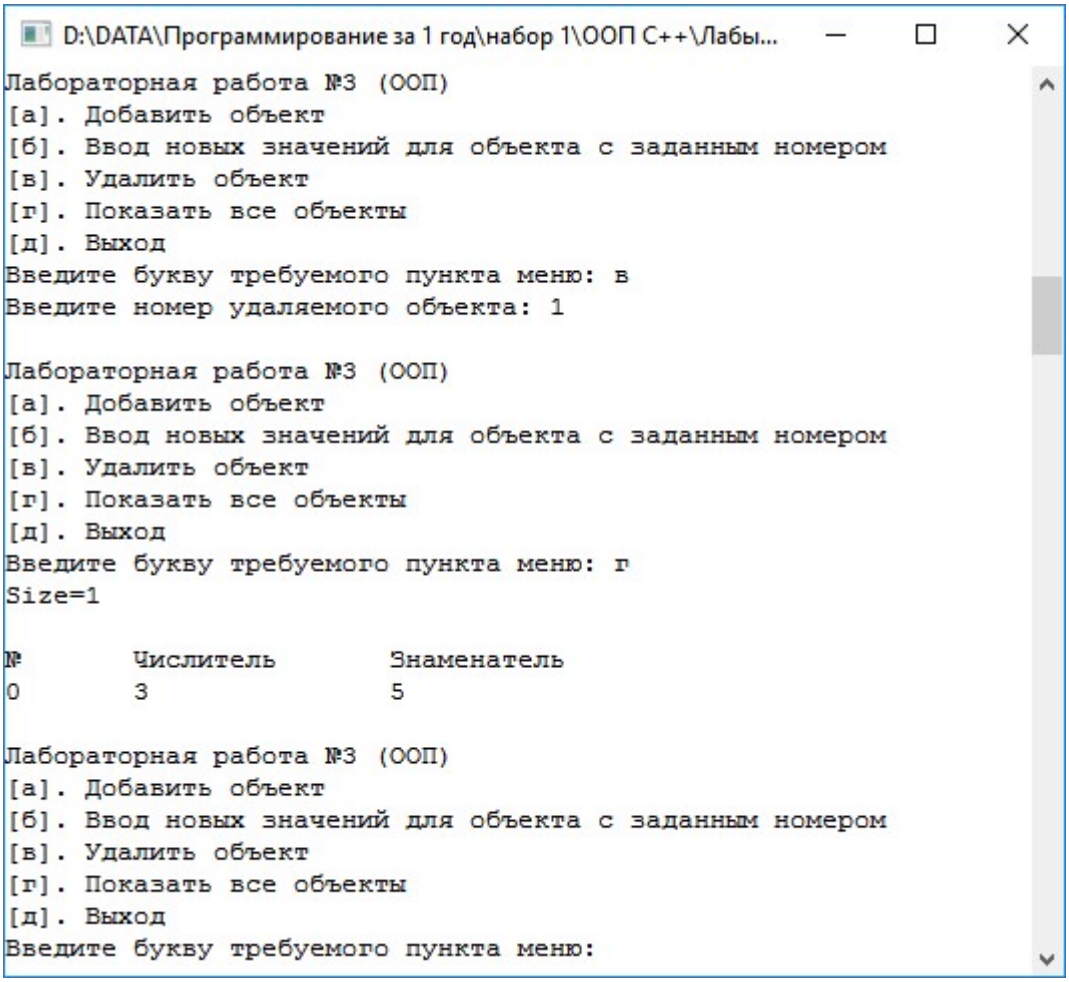

Рис. 7 Результат выбора пункта в главного меню программы.

При выборе пункта д главного меню происходит очистка памяти из-под объектов и массива указателей и выход из программы.

5. Выводы

В результате выполнения данной лабораторной работы, получен практический опыт работы с абстрактными классами и виртуальными методами. Целью данной работы является знакомство с динамическим полиморфизмом.

В данной лабораторной работе создан базовый абстрактный класс с чисто виртуальными методами и виртуальным деструктором. В классе-наследнике определены данные и методы:

- static поле динамического массива указателей на объекты абстрактного базового класса.
- static поле количества созданных объектов класса-наследника.
- Конструкторы и деструктор.
- static методы работы с этими полями.
- Реализации чисто виртуальных функций базового класса.

В программе добавлена возможность применять к объектам класса следующие действия:

- Хранить указатели на объекты-дроби в static динамическом массиве внутри класса дробей;
- Добавлять в static динамический массив указатели на вновь созданные объекты;
- Удалять из static динамического массива указатели на удаленные объекты по индексу в массиве указателей;

- Изменять значения объектов по индексу в массиве указателей;
- Очищать память из-под объектов и массива указателей.

Файлы программы находятся в репозитории по адресу: https://github.com/broitman-eugeny/lab3.

6. Код программы с комментариями

“ Fraction.h ”

```
#pragma once
#include <iostream>
#include <Windows.h>
//Абстрактный базовый класс дроби
class AbstractFraction
{
public:
	virtual void Print() = 0;		//Вывод на экран (чисто виртуальная функция)
	virtual ~AbstractFraction(){}	//Виртуальный деструктор для исключения утечки памяти при удалении производного объекта по указателю базового класса
	virtual int & GetNumerator()=0;	//Извлечение значения числителя (чисто виртуальная функция)
	virtual int & GetDenominator()=0; //Извлечение значения знаменателя (чисто виртуальная функция)
};
//Класс дроби
class Fraction: public AbstractFraction
{
	int Numerator;			//Числитель
	int Denominator;	//Знаменатель
	static AbstractFraction **PFraction;//Static ДИНАМИЧЕСКИЙ массив указателей на объекты БАЗОВОГО (абстрактного) класса
	static int Size;	//Размер массива PFraction
public:
	Fraction();							//Конструктор по умолчанию
	Fraction(const int N, const int D);			//Конструктор с параметрами
	Fraction(const Fraction &F);				//Конструктор копирования
	virtual ~Fraction();			//Виртуальный деструктор для исключения утечки памяти при удалении производного объекта по указателю базового класса
	static void SetNumerator(const int N, const int Index);		//Установка значения числителя объекта, на который ссылается указатель массива PFraction по индексу Index
	static void SetDenominator(const int D, const int Index);	//Установка значения знаменателя объекта, на который ссылается указатель массива PFraction по индексу Index
	virtual int & GetNumerator();		//Извлечение значения числителя (унаследованная чисто виртуальная функция)
	virtual int & GetDenominator();		//Извлечение значения знаменателя (унаследованная чисто виртуальная функция)
	static int GetSize();				//Извлечение размера массива PFraction
	virtual void Print();				//Вывод на экран (унаследованная чисто виртуальная функция)
	static void PrintAll();				//Вывод на экран всех объектов, указатели на которые находятся в PFraction
	static void clearArray();		//Удаление всех объектов, на которые ссылаются указатели массива PFraction
	static void delElem(const int n);		//Удаление объекта, на который ссылается указатель массива PFraction по индексу n
};
//Функция меню
void Menu();
```

“Fraction.cpp”

```
#include "Fraction.h"
//Инициализация статических членов класса Fraction
int Fraction::Size = 0;
AbstractFraction ** Fraction::PFraction = NULL;
//Конструктор по умолчанию
Fraction::Fraction():Numerator(1), Denominator(1)
{
	//Создание нового массива указателей с кол-вом эл-тов на 1 больше, чем в PFraction
	AbstractFraction** nArr = new AbstractFraction*[Size + 1];
	//Скопировать все эл-ты PFraction в новый массив
	for (int i = 0; i < Size; i++)
	{
		nArr[i] = PFraction[i];
	}
	nArr[Size++] = this;//Вновь созданный элемент тоже записать в массив
	if (PFraction != NULL)
	{
```

```
			delete[] PFraction;
		}
		PFraction = nArr;//Вновь созданный массив - член класса Fraction
}
//Конструктор с параметрами
Fraction::Fraction(const int N, const int D) : Numerator(N), Denominator(D)
{
		//Создание нового массива указателей с кол-вом эл-тов на 1 больше, чем в PFraction
		AbstractFraction** nArr = new AbstractFraction*[Size + 1];
		//Скопировать все эл-ты PFraction в новый массив
		for (int i = 0; i < Size; i++)
		{
			nArr[i] = PFraction[i];
		}
		nArr[Size++] = this;//Вновь созданный элемент тоже записать в массив
		if (PFraction != NULL)
		{
			delete[] PFraction;
		}
		PFraction = nArr;//Вновь созданный массив - член класса Fraction
}
//Конструктор копирования
Fraction::Fraction(const Fraction & F) : Numerator(F.Numerator), Denominator(F.Denominator)
{
		//Создание нового массива указателей с кол-вом эл-тов на 1 больше, чем в PFraction
		AbstractFraction** nArr = new AbstractFraction*[Size + 1];
		//Скопировать все эл-ты PFraction в новый массив
		for (int i = 0; i < Size; i++)
		{
			nArr[i] = PFraction[i];
		}
		nArr[Size++] = this;//Вновь созданный элемент тоже записать в массив
		if (PFraction != NULL)
		{
			delete[] PFraction;
		}
		PFraction = nArr;//Вновь созданный массив - член класса Fraction
}
//Деструктор
Fraction::~Fraction()
{
		if (Size == 1)
		{
			delete [] PFraction;//удаляется массив указателей, сам элемент удаляется в функциях
clearArray и delElem после отработки деструктора
			PFraction = NULL;
			Size--;
		}
		else
			if(Size > 1 && PFraction[Size - 1] != this)//если удаляем элемент не с максимальным
значением индекса в массиве
			{
				AbstractFraction** nArr = new AbstractFraction*[Size - 1];
				for (int i = 0, j = 0; i < Size; i++)
				{
					if (PFraction[i] != this)//Копируются все, кроме указателя на удаляемый
элемент
					{
						nArr[j++] = PFraction[i];
					}
				}
				delete[] PFraction;//Удаление массива указателей
				PFraction = nArr;//Вновь созданный массивn Arr - член класса Fraction
				Size--;
			}
			else
				if (Size > 1 && PFraction[Size - 1] == this)//если удаляем элемент с максималь-
ным значением индекса в массиве
				{
```

```
                        Size--;
                }
}
//Установка значения числителя объекта
void Fraction::SetNumerator(const int N, const int Index)
{
	PFraction[Index]->GetNumerator() = N;
}
//Установка значения знаменателя
void Fraction::SetDenominator(const int D, const int Index)
{
	PFraction[Index]->GetDenominator() = D;
}
//Извлечение ссылки на числитель (виртуальная реализация)
int & Fraction::GetNumerator()
{
	return Numerator;
}
//Извлечение ссылки на знаменатель (виртуальная реализация)
int & Fraction::GetDenominator()
{
	return Denominator;
}
int Fraction::GetSize()
{
	return Size;
}
//Вывод на экран (реализация виртуальной функции)
void Fraction::Print()
{
	std::cout << Numerator << "\t\t" << Denominator << std::endl;
}
//Вывод на экран всех объектов на которые ссылаются указатели массива PFraction
void Fraction::PrintAll()
{
	std::cout << "Size=" << Size << std::endl;
	std::cout << std::endl << "№" << "\t" << "Числитель" << "\t" << "Знаменатель" << std::endl;
	for (int i = 0; i < Size; i++)
	{
		std::cout << i << "\t";
		PFraction[i]->Print();
	}
}
//Удаление всех объектов, на которые ссылаются указатели массива PFraction
void Fraction::clearArray()
{
	for (int i= Size-1;i>=0;i--)
	{
		delete PFraction[i];
	}
}
//Удаление объекта на который ссылается указатель по индексу n в массиве PFraction
void Fraction::delElem(const int n)
{
	if (n >= 0 && n < Size)
	{
		delete PFraction[n];
	}
}
```

“Main.cpp”

```
//Лабораторная работа №3.
//Вариант №5.
//1) Разработать структуру абстрактного класса, который объявляет собой минимально необходимый ин-
терфейс.
//Минимальный интерфейс включает в себя функцию вывода.
//2) Разработать производный класс (согласно варианту задания по работам №1, №2),
//осуществив его наследование от разработанного абстрактного класса, с реализацией виртуальной функ-
ции.
//Методы, добавленные в работе №2 (перегружающие операторы), в данный класс не включать.
```

```
//Класс должен включить в себя :
//-    Поля - данные, для хранения значений.
//-    Реализацию виртуальной функции вывода.
//-    Static ДИНАМИЧЕСКИЙ массив указателей на объекты БАЗОВОГО (абстрактного) класса.
//-    Статическую переменную для хранения размера массива.
//-    Статические функции:
//     -      добавления указателя(типа Abstract****) на объект данного класса.
//     -      удаление элемента с заданным номером.
//     -      обход всего массива с вызовом функции вывода.
#include "Fraction.h"
void main()
{
	SetConsoleCP(1251);//Ввод русских букв
	SetConsoleOutputCP(1251);//Вывод русских букв
	Menu();
}
```

“Menu.cpp”

```
#include "Fraction.h"
//Функция отображения меню
void ShowMenu()
{
	std::cout << std::endl << "Лабораторная работа №3 (ООП)";
	std::cout << std::endl << "[а]. Добавить объект";
	std::cout << std::endl << "[б]. Ввод новых значений для объекта с заданным номером";
	std::cout << std::endl << "[в]. Удалить объект";
	std::cout << std::endl << "[г]. Показать все объекты";
	std::cout << std::endl << "[д]. Выход";
	std::cout << std::endl << "Введите букву требуемого пункта меню: ";
}
//Функция отображения подменю добавления объекта
void ShowMenu2()
{
	std::cout << std::endl << "[1]. Значения по умолчанию";
	std::cout << std::endl << "[2]. Ввод значений числителя и знаменателя";
	std::cout << std::endl << "[3]. Выход";
	std::cout << std::endl << "Введите номер требуемого пункта подменю: ";
}
//Функция проверки ввода из потока std::cin целочисленного значения
//Value - ссылка на переменную в которую осуществляется ввод
//LeftBorder - левая граница области определения вводимой переменной
//LeftIncluded - признак включения в область определения переменной левой границы
//RightBorder - правая граница области определения вводимой переменной
//RightIncluded - признак включения в область определения переменной правой границы
void CheckCinInt(int &Value, const int LeftBorder, const bool LeftIncluded, const int RightBorder,
const bool RightIncluded)
{
	while ((LeftIncluded==true)?(Value<LeftBorder):(Value<=LeftBorder) || (RightIncluded == true)
? (Value>RightBorder) : (Value >= RightBorder) || std::cin.fail())
	{
		std::cin.clear();
		std::cin.ignore(std::cin.rdbuf()->in_avail());//Очистка буфера
		std::cout << std::endl << "Значение введено неверно, введите еще раз: ";
		std::cin >> Value;
	}
}
//Функция проверки ввода из потока std::cin значения типа char
//Value - ссылка на переменную в которую осуществляется ввод
//LeftBorder - левая граница области определения вводимой переменной
//LeftIncluded - признак включения в область определения переменной левой границы
//RightBorder - правая граница области определения вводимой переменной
//RightIncluded - признак включения в область определения переменной правой границы
void CheckCinChar(char &Value, const char LeftBorder, const bool LeftIncluded, const char
RightBorder, const bool RightIncluded)
{
	while ((LeftIncluded == true) ? (Value<LeftBorder) : (Value <= LeftBorder) || (RightIncluded
== true) ? (Value>RightBorder) : (Value >= RightBorder) || std::cin.fail())
	{
		std::cin.clear();
		std::cin.ignore(std::cin.rdbuf()->in_avail());//Очистка буфера
```

```
			std::cout << std::endl << "Значение введено неверно, введите еще раз: ";
			std::cin >> Value;
		}
	}
	//Функция меню
	void Menu()
	{
		char C = -1;
		int Index, C2;
		while (C != 'д')//д - выход
		{
			ShowMenu();
			std::cin >> C;
			CheckCinChar(C, 'а', true, 'д', true);//проверка корректности ввода
			switch (C)
			{
			case 'а'://Добавить объект
				C2 = -1;
				while (C2 != 3)//3 - выход
				{
					ShowMenu2();
					std::cin >> C2;
					CheckCinInt(C2, 1, true, 3, true);//проверка корректности ввода
					switch (C2)
					{
					case 1://Значения по умолчанию
						new Fraction();
						break;
					case 2://Ввод значений числителя и знаменателя
						std::cout << "Числитель: ";
						int N, D;
						std::cin >> N;
						CheckCinInt(N, INT_MIN, true, INT_MAX, true);//проверка корректно-
сти ввода
						std::cout << "Знаменатель: ";
						std::cin >> D;
						CheckCinInt(D, INT_MIN, true, INT_MAX, true);//проверка корректно-
сти ввода
						new Fraction(N, D);
						break;
					}//switch (C2)
				}//while (C2 != 3)//3 - выход
				break;
			case 'б'://Ввод новых значений для объекта с заданным номером
				std::cout << "Введите номер объекта: ";
				std::cin >> Index;
				if (Index < 0 || Index >= Fraction::GetSize())
				{
					std::cout << "Отсутствует элемент с указанным номером" << std::endl;
					break;
				}
				std::cout << "Числитель: ";
				int N, D;
				std::cin >> N;
				CheckCinInt(N, INT_MIN, true, INT_MAX, true);//проверка корректности ввода
				std::cout << "Знаменатель: ";
				std::cin >> D;
				CheckCinInt(D, INT_MIN, true, INT_MAX, true);//проверка корректности ввода
				Fraction::SetNumerator(N, Index);
				Fraction::SetDenominator(D, Index);
				break;
			case 'в'://Удалить объект
				if (Fraction::GetSize() == 0)
				{
					std::cout << "Сначала добавьте объекты" << std::endl;
					break;
				}
				std::cout << "Введите номер удаляемого объекта: ";
				std::cin >> Index;
```

```
                        if (Index < 0 || Index >= Fraction::GetSize())
                        {
                                std::cout << "Отсутствует элемент с указанным номером" << std::endl;
                                break;
                        }
                        Fraction::delElem(Index);
                        break;
                case 'г'://Показать все объекты
                        Fraction::PrintAll();
                        break;
                case 'д'://Выход
                        Fraction::clearArray();
                        break;
                }//switch (C)
        }//while (C != 'д')//д - выход
}
```