

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

Учебный Центр Информационных Технологий «Информатика»



Лабораторная работа № 1
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

Реализовать в заголовочном файле интерфейс класса.

Исходя из варианта задания, включить в класс необходимые данные-члены, конструкторы, деструктор, и остальные методы.

Класс должен включать в себя:

- Данные - члены, типы и количество которых хорошо подходят для хранения и представления соответствующих значений.
- Конструктор по умолчанию, конструктор с параметрами (возможно, не один), конструктор копирования и деструктор.
- Методы установки значений и вывода значений на экран (использовать потоковый ввод / вывод).
- Static - счетчик количества существующих объектов класса.
- В подходящих для этого методах, использовать параметры по умолчанию.
- Использовать в конструкторах список инициализации.

В файле методов реализовать интерфейс класса.

Класс хранит значение дроби. Предусмотреть метод сокращения дроби, умножения и деления на другую дробь (передавать как параметры функции).

2. Структурное описание

В данной лабораторной работе мы реализуем класс **Fraction**. Для удобства будем называть его «класс».

Для реализации данного класса созданы пять файлов. Заголовочный файл и четыре файла исходного кода.

В заголовочном файле “Fraction.h” описывается интерфейс (тело) класса, в файле “Fraction.cpp” реализованы методы класса, в файле “Main.cpp” размещена функция void main() в которой создается динамический массив указателей на объекты класса, вызывается функция пользовательского меню, и после работы с массивом – очистка памяти из под него, в файле “Menu.cpp” реализована функция пользовательского меню, которая обеспечивает пользовательский интерфейс класса, и в файле “Evklid.cpp” реализована функция поиска наибольшего общего делителя для 2-ух чисел.

Класс содержит в себе закрытые (private) члены класса: целочисленные переменные **Numerator** (числитель), **Denominator** (знаменатель) и статическую переменную **Count** (подсчитывает количество созданных объектов).

В открытые (public) члены класса входят конструктор класса по умолчанию **Fraction()**, конструктор с параметрами **Fraction(int N, int D)**, конструктор копирования **Fraction(Fraction &F)**, деструктор **~Fraction()**, метод **void Reduction()** - выполняет сокращение дроби, метод **void Mul(Fraction &F)** – выполняет умножение дроби на дробь, переданную в качестве параметра, метод **void Div(Fraction &F)** - выполняет деление дроби на дробь, переданную в качестве параметра, метод **void SetNumerator(int N)** – устанавливает значение числителя дроби, метод **void SetDenominator(int D)** - устанавливает значение знаменателя дроби, метод **int GetNumerator()** – возвращает значение числителя дроби, метод **int GetDenominator()** - возвращает значение знаменателя дроби, метод **static int GetCount()** - возвращает значение счетчика **Count**, метод **void Print()** – выводит на экран значения числителя и знаменателя дроби.

3. Функциональное описание

Рассмотрим реализацию методов класса.

1. Подключаем наш заголовочный файл «Fraction.h».
2. Инициализируем статическую переменную **Count = 0**.
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**. Статическая переменная **Count** увеличивается на **1**.
4. Конструктор с параметрами **Fraction (int N, int D)** – запускается автоматически при создании нового объекта класса с параметрами, принимает в качестве параметров 2 целочисленных значения, этими значениями инициализирует закрытые поля класса **Numerator** и **Denominator**. Статическая переменная счетчик **Count** увеличивается на **1**.
5. Конструктор копирования **Fraction(Fraction & F)** – используется при создании нового объекта класса, путем копирования существующего объекта. В качестве параметра принимает ссылку на существующий объект класса и создает его копию, выполняется копирование полей **Numerator** и **Denominator**. Статическая переменная счетчик **Count** увеличивается на **1**.
6. Деструктор **~Fraction ()** – запускается автоматически при удалении объекта класса. Статическая переменная счетчик **Count** уменьшается на **1**.
7. **void Reduction()** – метод выполняет сокращение дроби. В теле метода инициализируется целочисленная переменная **int CommonDevider**. В эту переменную записывается результат работы метода **int Evklid(int a, int b)**, т.е. наибольший общий делитель далее **НОД**. Выполняется сокращение числителя и знаменателя, делением на **НОД**.
8. **void Mul(Fraction & F)** - метод выполняет умножение дроби на дробь, ссылка на которую передается в качестве параметра. Выполняется умножение числителя и знаменателя соответственно на числитель и знаменатель объекта, переданного по ссылке.
9. **void Div (Fraction & F)** - метод выполняет деление дроби на дробь, ссылка на которую передается в качестве параметра. Выполняется умножение числителя на знаменатель объекта, переданного по ссылке, и знаменателя на числитель объекта, переданного по ссылке.
10. **void SetNumerator(int N)** - метод устанавливает значение числителя, в качестве параметра принимает новое целочисленное значение числителя.
11. **void SetDenominator(int D)** - метод устанавливает значение знаменателя, в качестве параметра принимает новое целочисленное значение знаменателя.
12. **int GetNumerator()** - метод не принимает никаких параметров, в качестве результата возвращает целочисленное значение числителя.
13. **int GetDenominator()** - метод не принимает никаких параметров, в качестве результата возвращает целочисленное значение знаменателя.

14. **static int GetCount()** – метод не принимает никаких параметров, в качестве результата возвращает целочисленное значение счетчика **Count**.

15. **void Print()** – метод вывода объекта на экран. Выводит на экран через табуляцию значения числителя и знаменателя дроби.

4. Описание работы программы

При запуске программы пользователю предлагается выбрать пункт меню первого уровня. Рис. 1.

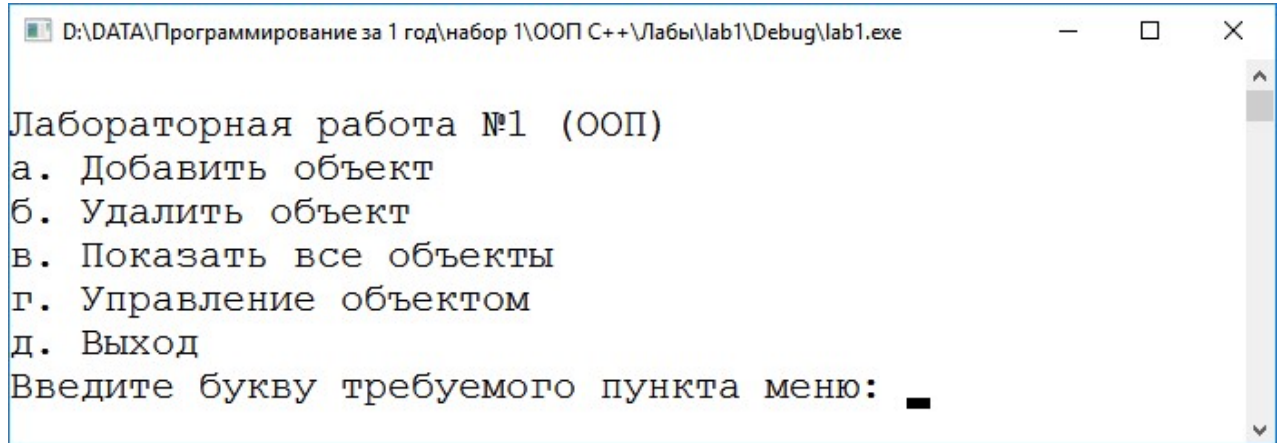


Рис. 1 Меню первого уровня.

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

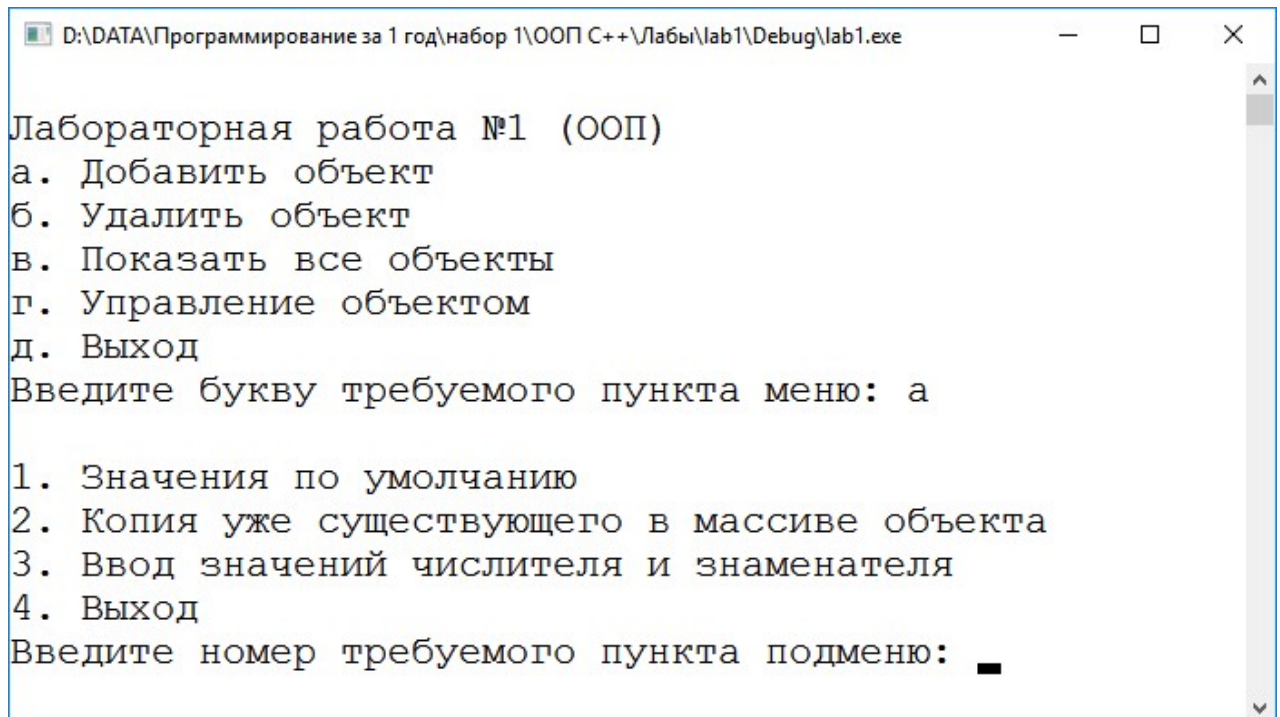


Рис. 2 Меню второго уровня добавления объекта.

При выборе пункта 1 меню второго уровня добавления объекта будет создан объект **Fraction** при помощи конструктора по умолчанию и на экран будет выведено сообщение об общем количестве созданных объектов и повторно меню второго уровня добавления объекта. Рис. 3.

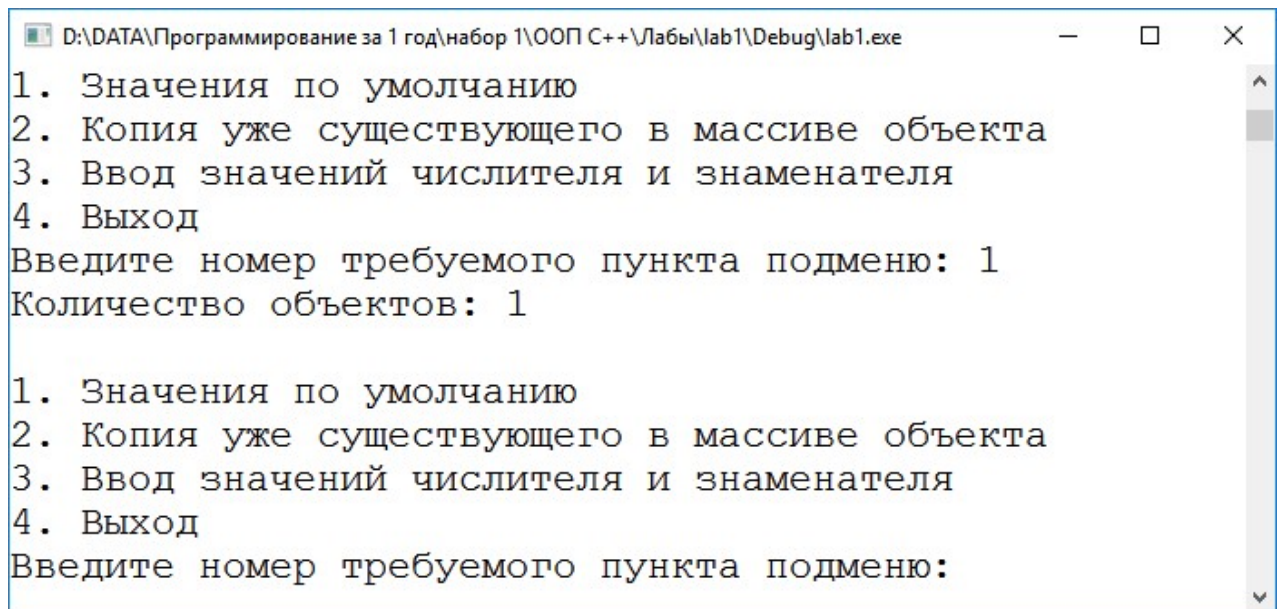


Рис. 3 Результат выбора пункта 1 меню второго уровня добавления объекта.

При выборе пункта 2 меню второго уровня добавления объекта будет предложено ввести индекс копируемого объекта. Рис. 4.

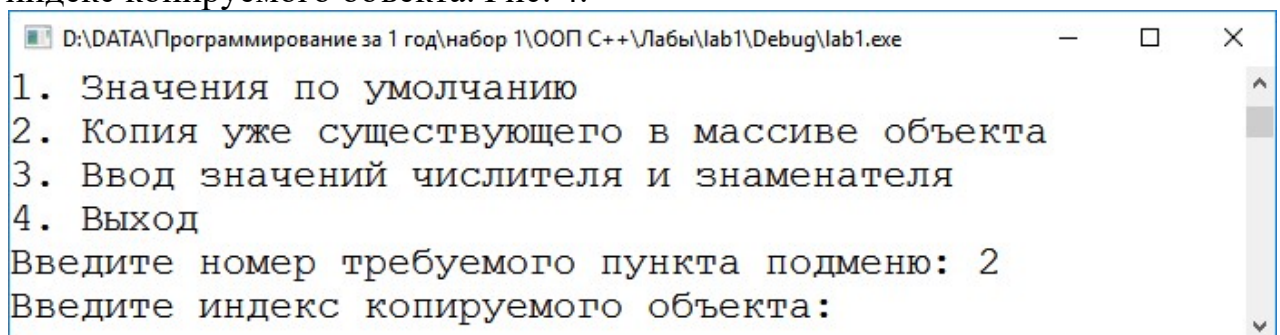


Рис. 4 Результат выбора пункта 2 меню второго уровня добавления объекта.

После ввода индекса копируемого объекта будет создан объект Fraction при помощи конструктора копирования и на экран будет выведено сообщение об общем количестве созданных объектов и повторно меню второго уровня добавления объекта. Рис. 5.

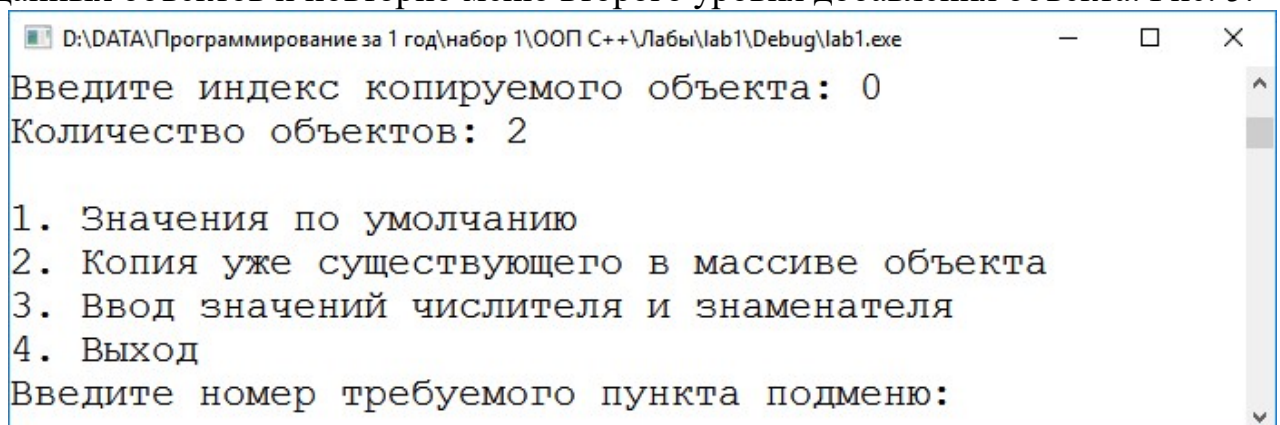


Рис. 5 Результат ввода индекса копируемого объекта.

При выборе пункта 3 меню второго уровня добавления объекта будет предложено ввести значения числителя и знаменателя дроби. Рис. 6.

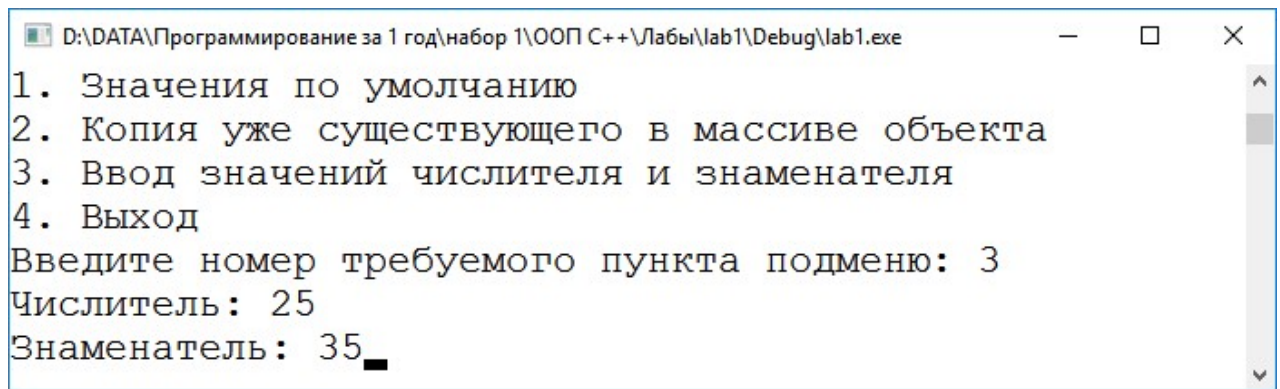


Рис. 6 Результат выбора пункта 3 меню второго уровня добавления объекта.

После ввода значений числителя и знаменателя дроби будет создан объект Fraction при помощи конструктора с параметрами и на экран будет выведено сообщение об общем количестве созданных объектов и повторно меню второго уровня добавления объекта. Рис. 7.

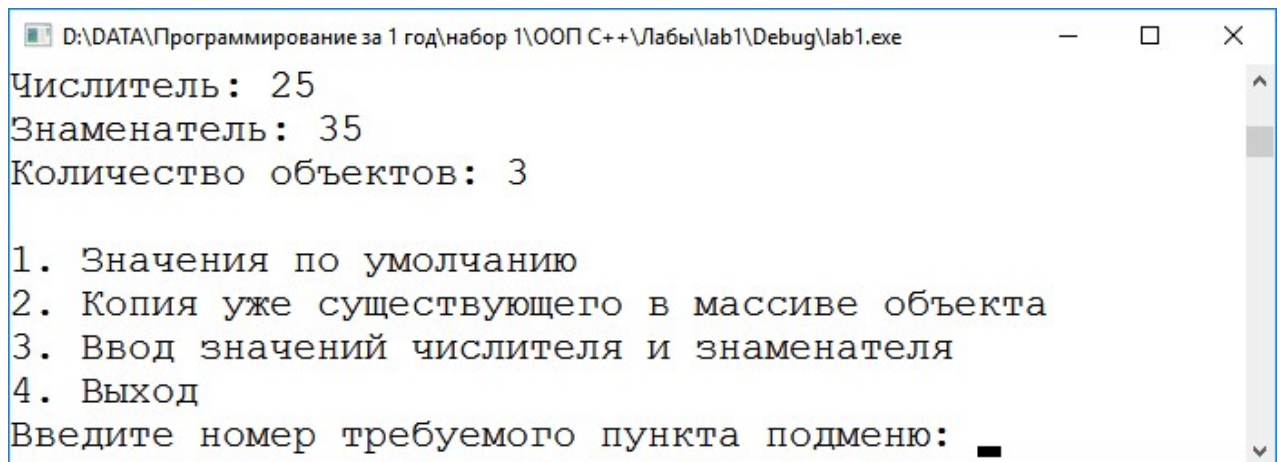


Рис. 7 Результат ввода значений числителя и знаменателя дроби.

При выборе пункта 4 меню второго уровня добавления объекта на экран будет выведено меню первого уровня. Рис. 8.

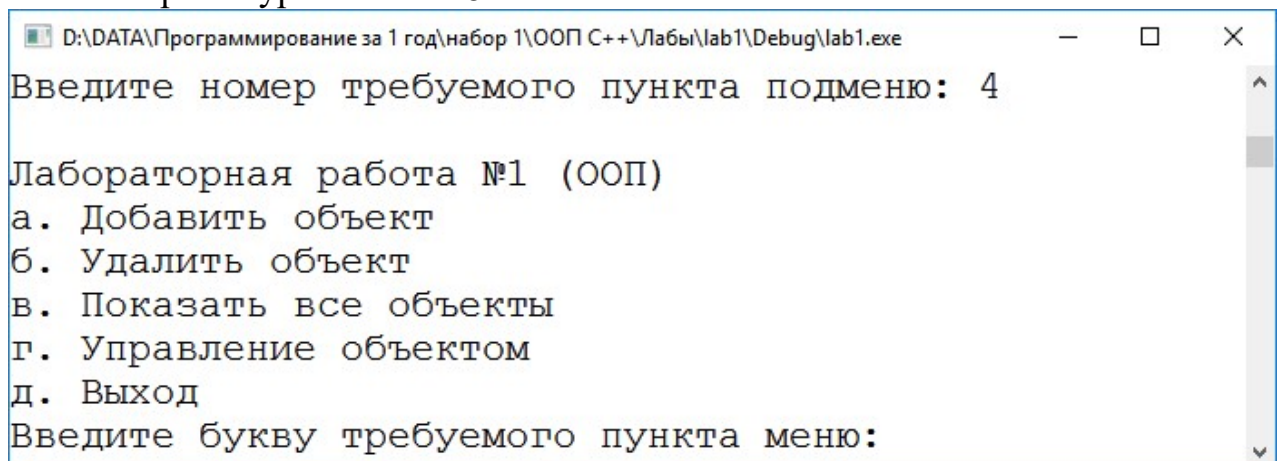


Рис. 8 Результат выбора пункта 4 меню второго уровня добавления объекта.

При выборе пункта б на экран будет предложено ввести индекс удаляемого объекта. Рис. 9.

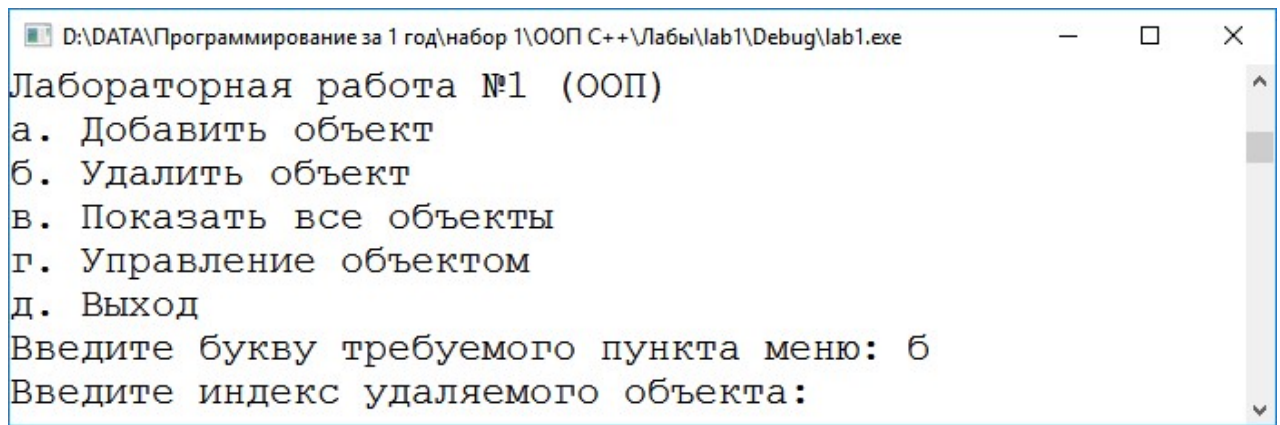


Рис. 9 Меню второго уровня добавления объекта.

После ввода значения индекса удаляемого объекта будет удален объект Fraction, следующие за ним объекты будут сдвинуты на одну позицию назад, и на экран будет выведено сообщение об общем количестве созданных объектов и повторно меню первого уровня. Рис. 10.

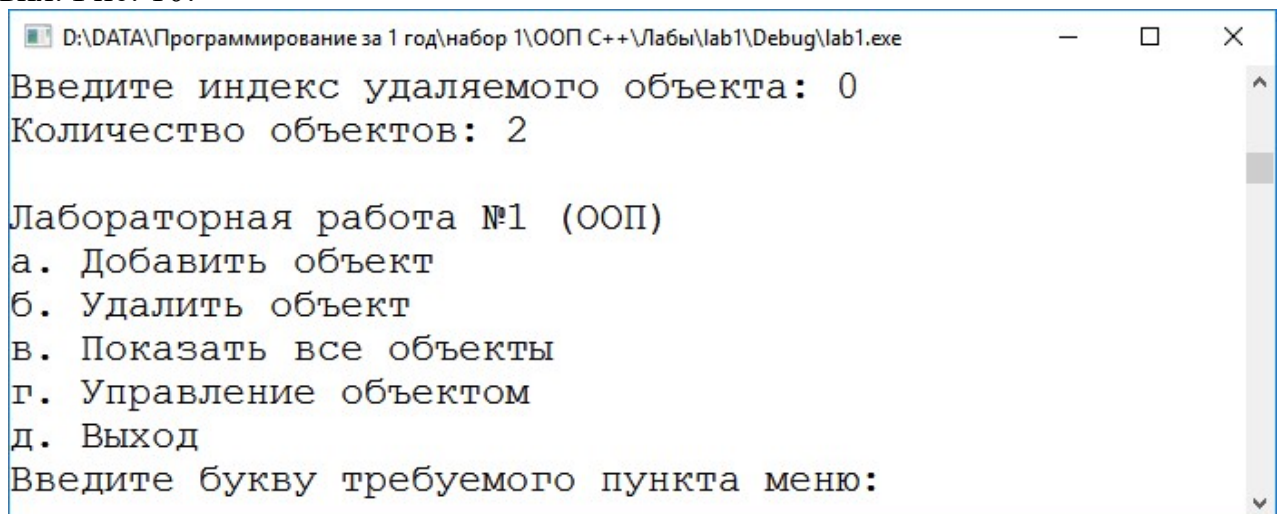


Рис. 10 Результат ввода значения индекса удаляемого объекта.

При выборе пункта в на экран будут выведены в табличном виде все объекты. Рис. 11.

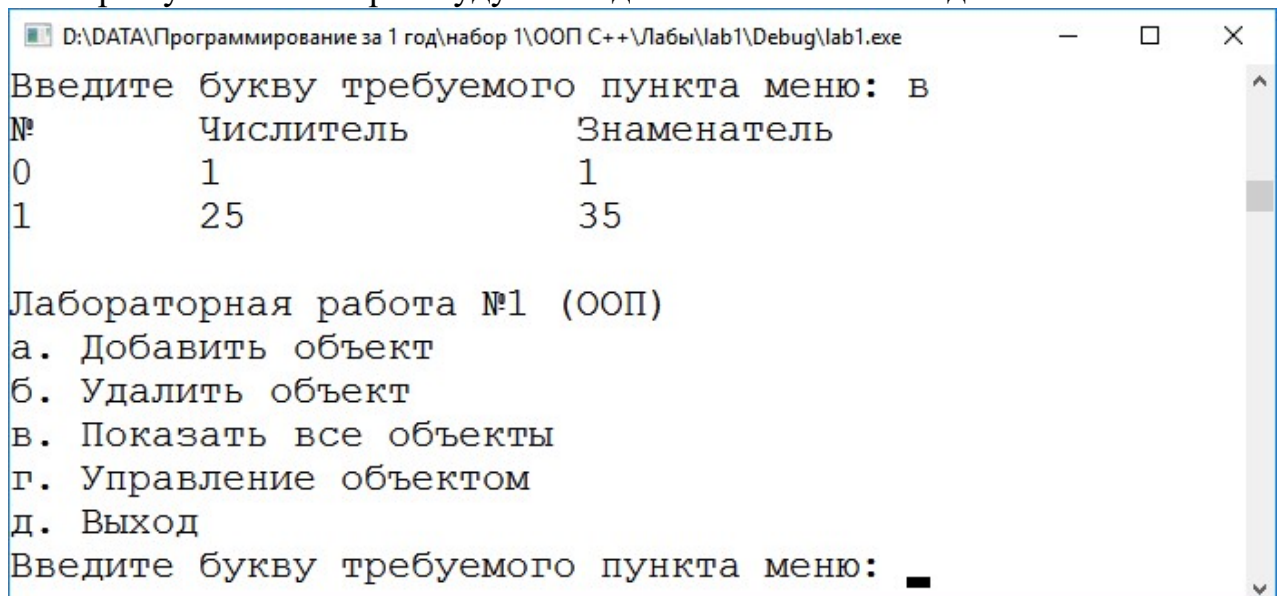


Рис. 11 Результат выбора пункта в меню первого уровня.

При выборе пункта г будет предложено ввести индекс управляемого объекта. Рис. 12.

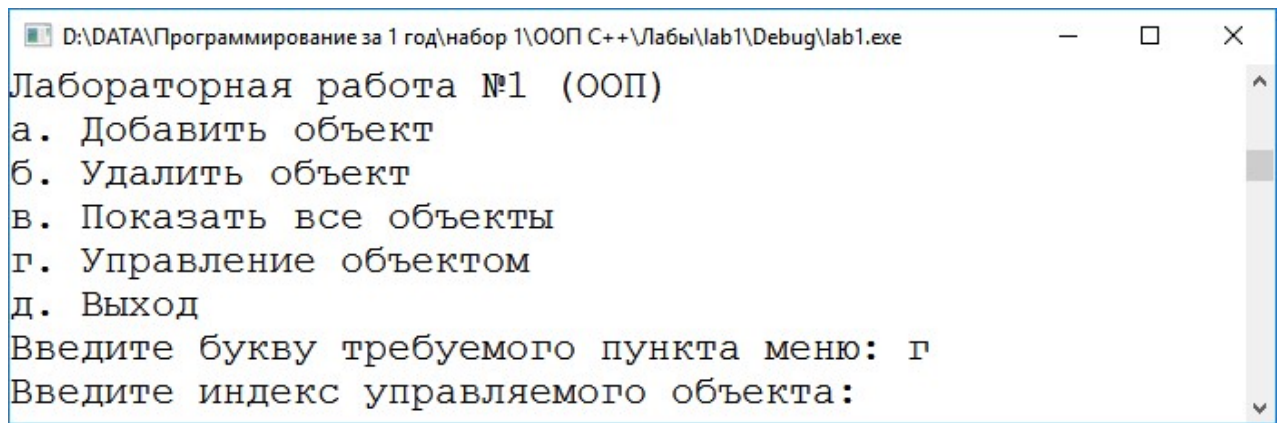


Рис. 12 Результат выбора пункта г меню первого уровня.

После ввода индекса управляемого объекта на экран будет выведено меню второго уровня управления объектом. Рис. 13.

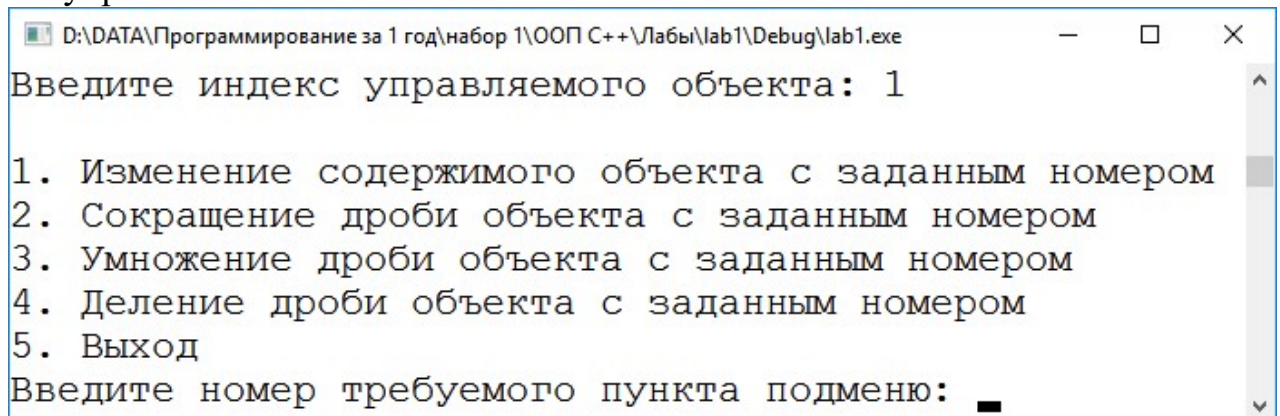


Рис. 13 Результат ввода индекса управляемого объекта.

При выборе пункта 1 меню второго уровня управления объектом будет предложено ввести новые значения числителя и знаменателя дроби. Рис. 14.

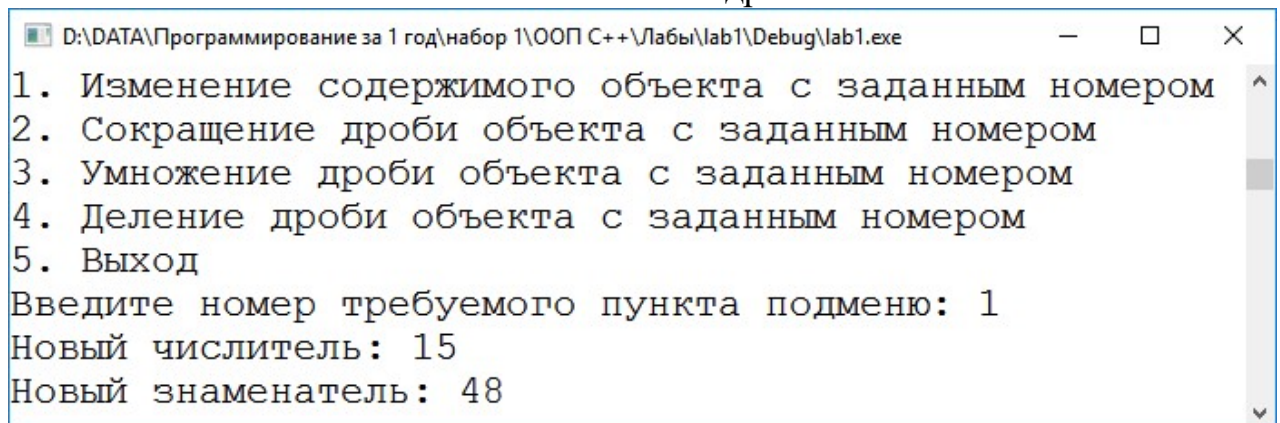


Рис. 14 Результат выбора пункта 1 меню второго уровня управления объектом.

После ввода новых значений числителя и знаменателя дроби объект будет обновлен и на экран будет выведено повторно меню второго уровня управления объектом. Рис. 15.

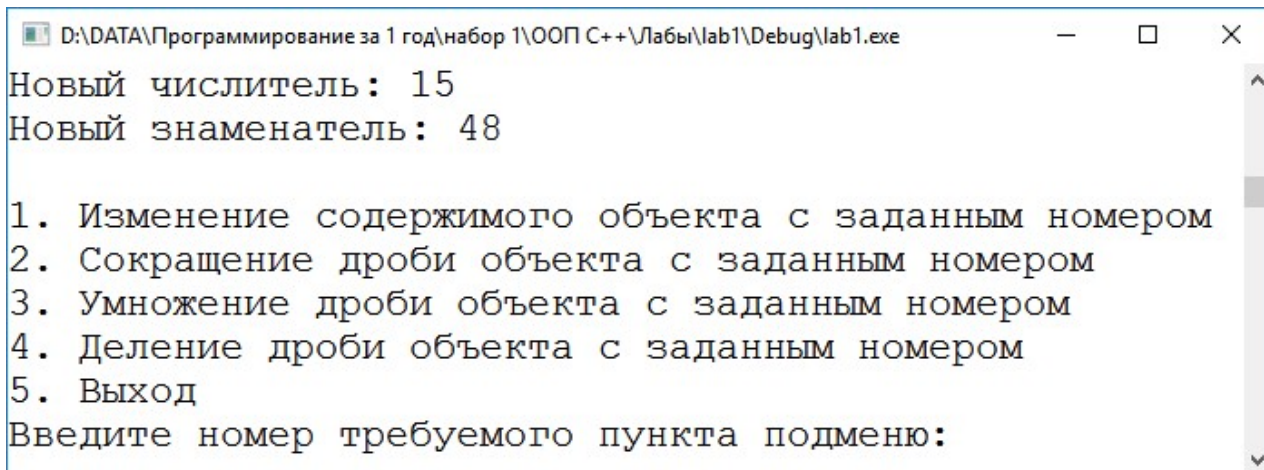


Рис. 15

Результат ввода новых значений числителя и знаменателя дроби.

При выборе пункта 2 меню второго уровня управления объектом дробь будет сокращена, и на экран будет выведено повторно меню второго уровня управления объектом. Рис. 16.

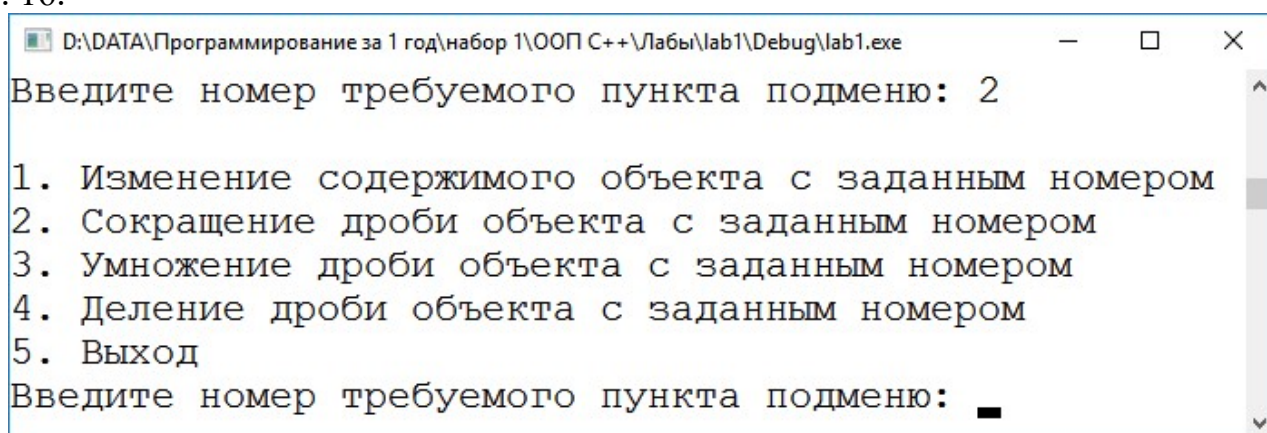


Рис. 16 Результат выбора пункта 2 меню второго уровня управления объектом.

При выборе пункта 3 меню второго уровня управления объектом будет предложено ввести значения числителя и знаменателя множителя. Рис. 17.

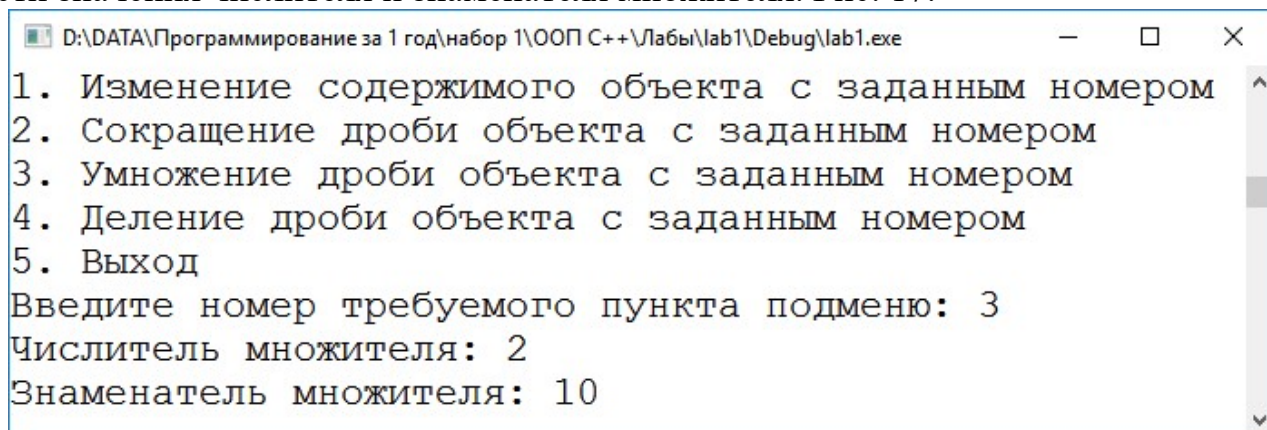


Рис. 17 Результат выбора пункта 3 меню второго уровня управления объектом.

После ввода значений числителя и знаменателя множителя объект будет умножен на множитель и обновлен, и на экран будет выведено повторно меню второго уровня управления объектом. Рис. 18.

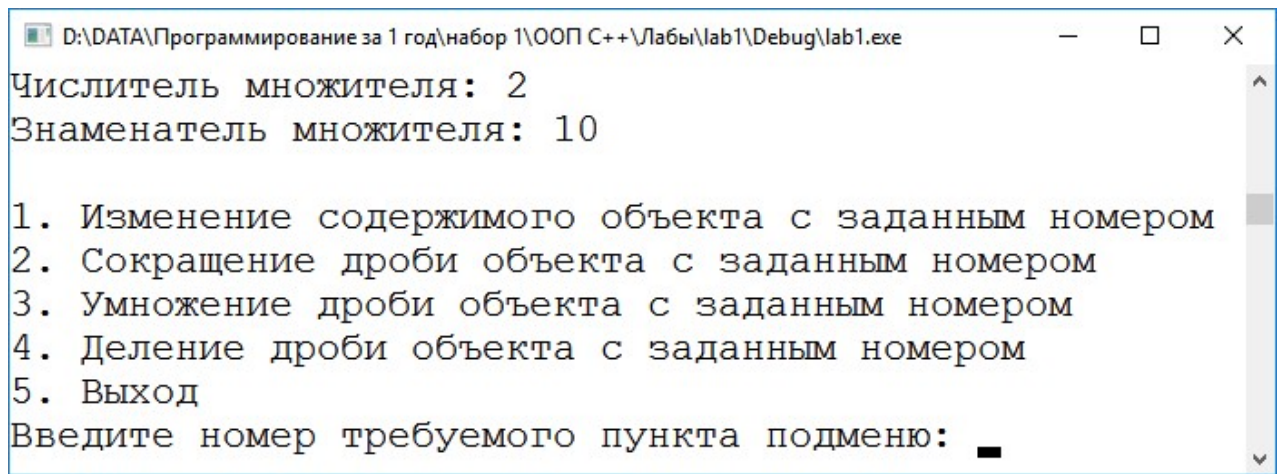


Рис. 18 Результат ввода значений числителя и знаменателя множителя.

При выборе пункта 4 меню второго уровня управления объектом будет предложено ввести значения числителя и знаменателя делителя. Рис. 19.

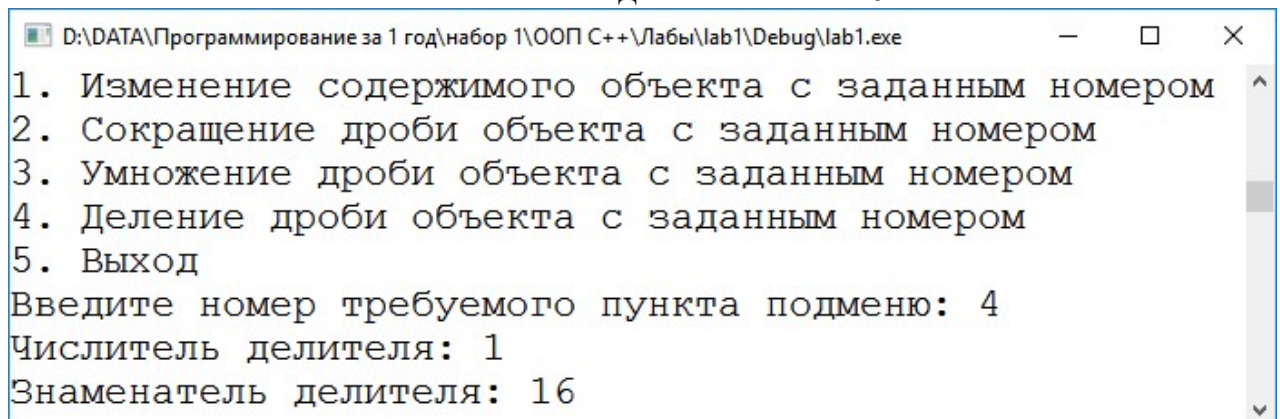


Рис. 19 Результат выбора пункта 4 меню второго уровня управления объектом.

После ввода значений числителя и знаменателя делителя объект будет поделен на делитель и обновлен, и на экран будет выведено повторно меню второго уровня управления объектом. Рис. 20.

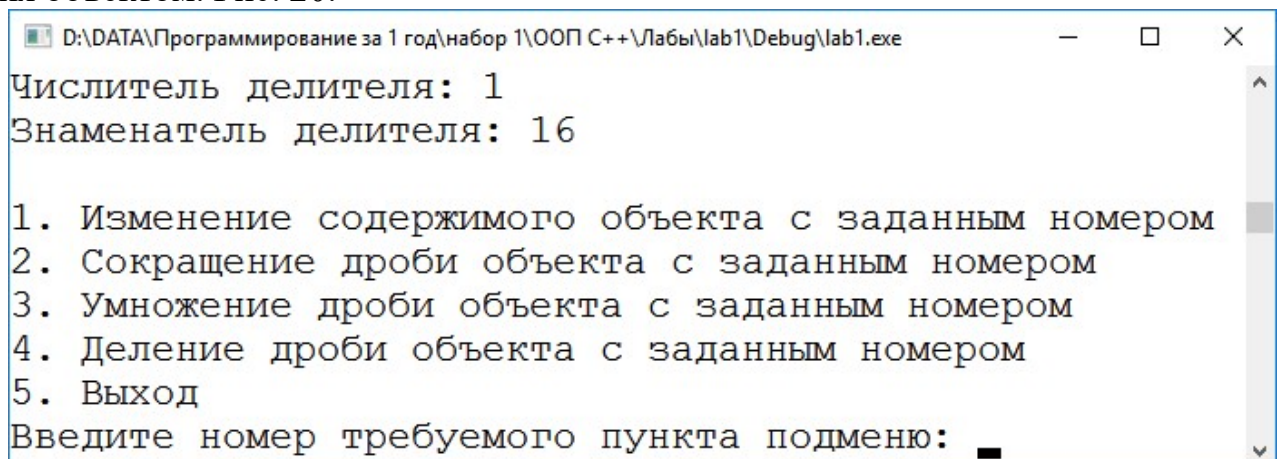


Рис. 20 Результат ввода значений числителя и знаменателя делителя.

При выборе пункта 5 меню второго уровня управления объектом на экран будет выведено меню первого уровня. Рис. 21.

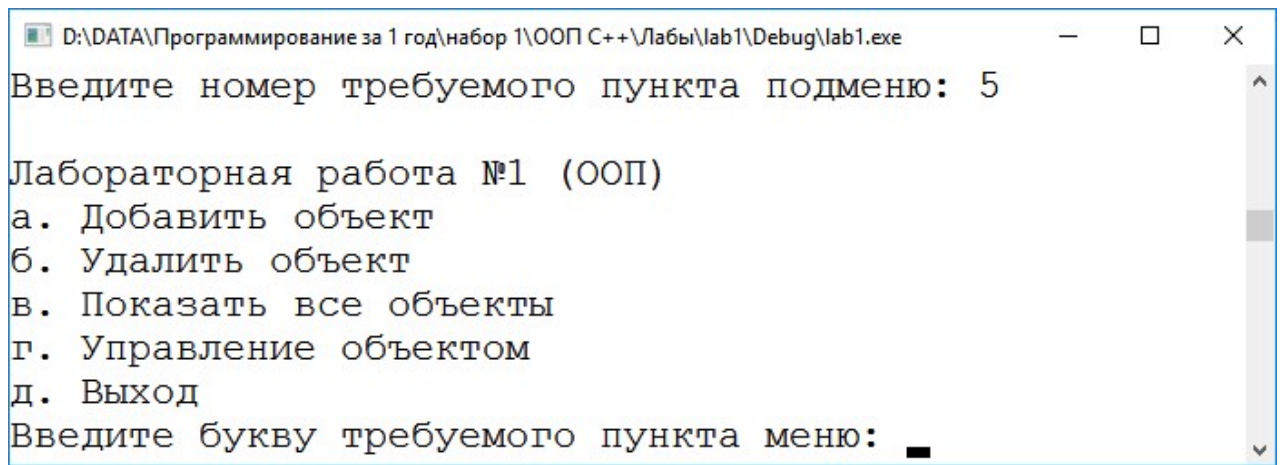


Рис. 21 Результат выбора пункта 5 меню второго уровня управления объектом.

При выборе пункта д меню первого уровня происходит выход из программы.

5. Выводы

В результате выполнения данной лабораторной работы, получен практический опыт в проектировании и реализации класса в ООП. Целью данной работы является знакомство с ООП на языке С++ и классами.

В данной лабораторной работе реализован класс «Дроби». В программе реализована возможность применять к объектам класса следующие действия:

- Создание дроби со значениями числителя и знаменателя по умолчанию.
- Создание дроби с произвольными значениями числителя и знаменателя.
- Создание дроби со значениями числителя и знаменателя из другого объекта класса «Дроби».
- Удаление объекта.
- Сокращение дроби.
- Умножение дроби на другую дробь.
- Деление дроби на другую дробь.
- Установка значения числителя.
- Установка значения знаменателя.
- Определение значения числителя.
- Определение значения знаменателя.
- Определение количества созданных объектов.
- Вывод на экран.

Файлы программы находятся в репозитории по адресу: <https://github.com/broitman-eugeny/lab1>

6. Код программы с комментариями

“Fraction.h”

```
#pragma once
#include <iostream>
#include <Windows.h>
const int FractionDimSize = 10;
//Класс дроби
class Fraction
{
    int Numerator;           //Числитель
    int Denominator;         //Знаменатель
    static int Count;        //Счетчик объектов
public:
    Fraction();              //Конструктор по умолчанию
    Fraction(int N, int D);  //Конструктор с параметрами
    Fraction(Fraction &F);   //Конструктор копирования
    ~Fraction();             //Деструктор
    void Reduction();        //Сокращение дроби
    void Mul(Fraction &F);   //Умножение дроби
    void Div(Fraction &F);   //Деление дроби
    void SetNumerator(int N); //Установка значения числителя
    void SetDenominator(int D); //Установка значения знаменателя
    int GetNumerator();       //Извлечение значения числителя
    int GetDenominator();    //Извлечение значения знаменателя
    static int GetCount();    //Извлечение количества созданных объектов
    void Print();            //Вывод на экран
};
//Определение НОД по алгоритму Евклида
int Evklid(int a, int b);
//Функция меню
//PFraction - массив указателей на объекты типа Fraction, состоящий из FractionDimSize элементов
void Menu(Fraction **PFraction);
```

“Fraction.cpp”

```
#include "Fraction.h"
int Fraction::Count = 0;
Fraction::Fraction():Numerator(1), Denominator(1)
{
    Count++;
}
Fraction::Fraction(int N, int D) : Numerator(N), Denominator(D)
{
    Count++;
}
Fraction::Fraction(Fraction & F) : Numerator(F.Numerator), Denominator(F.Denominator)
{
    Count++;
}
Fraction::~~Fraction()
{
    Count--;
}
void Fraction::Reduction()//Сокращение дроби
{
    int CommonDevider= Evklid(Numerator, Denominator);//наибольший общий делитель по алгоритму
    Евклида
    Numerator /= CommonDevider;
    Denominator /= CommonDevider;
}
void Fraction::Mul(Fraction & F)//Умножение дроби
{
    Numerator *= F.Numerator;
    Denominator *= F.Denominator;
}
void Fraction::Div(Fraction & F)
{
    Numerator *= F.Denominator;
    Denominator *= F.Numerator;
}
```



```

void Fraction::SetNumerator(int N)
{
    Numerator = N;
}
void Fraction::SetDenominator(int D)
{
    Denominator = D;
}
int Fraction::GetNumerator()
{
    return Numerator;
}
int Fraction::GetDenominator()
{
    return Denominator;
}
int Fraction::GetCount()
{
    return Count;
}
void Fraction::Print()
{
    std::cout << Numerator << "\t\t" << Denominator << std::endl;
}

```

“Main.cpp”

```

//Лабораторная работа №1.
//Вариант №5.
//Реализовать в заголовочном файле интерфейс класса.
//Исходя из варианта задания, включить в класс необходимые данные-члены,
//конструкторы, деструктор, и остальные методы.
//Класс должен включать в себя :
//• Данные - члены, типы и количество которых хорошо подходят для хранения и представления соответствующих значений.
//• Конструктор по умолчанию, конструктор с параметрами(возможно, не один), конструктор копирования и деструктор.
//• Методы установки значений и вывода значений на экран(использовать потоковый ввод / вывод).
//• Static - счетчик количества существующих объектов класса.
//• В подходящих для этого методах, используйте параметры по умолчанию.
//• Используйте в конструкторах список инициализации.
//В файле методов реализовать интерфейс класса.
//Класс хранит значение дроби.Предусмотреть метод сокращения дроби,
//умножения и деления на другую дробь(передавать как параметры функции).
#include "Fraction.h"
void main()
{
    Fraction **PFraction=new Fraction *[FractionDimSize];//массив указателей из FractionDimSize
    элементов
    SetConsoleCP(1251);//Ввод русских букв
    SetConsoleOutputCP(1251);//Вывод русских букв
    Menu(PFraction);
    delete[] PFraction;//освобождение памяти из под массива указателей
}

```

“Menu.cpp”

```

#include "Fraction.h"
//Функция отображения меню
void ShowMenu()
{
    std::cout << std::endl << "Лабораторная работа №1 (ООП)";
    std::cout << std::endl << "а. Добавить объект";
    std::cout << std::endl << "б. Удалить объект";
    std::cout << std::endl << "в. Показать все объекты";
    std::cout << std::endl << "г. Управление объектом";
    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. Копия уже существующего в массиве объекта";
std::cout << std::endl << "3. Ввод значений числителя и знаменателя";
std::cout << std::endl << "4. Выход";
std::cout << std::endl << "Введите номер требуемого пункта подменю: ";
}
//Функция отображения подменю управления объектом
void ShowMenu3()
{
    std::cout << std::endl << "1. Изменение содержимого объекта с заданным номером";
    std::cout << std::endl << "2. Сокращение дроби объекта с заданным номером";
    std::cout << std::endl << "3. Умножение дроби объекта с заданным номером";
    std::cout << std::endl << "4. Деление дроби объекта с заданным номером";
    std::cout << std::endl << "5. Выход";
    std::cout << std::endl << "Введите номер требуемого пункта подменю: ";
}
//Функция меню
//PFraction - массив указателей на объекты типа Fraction, состоящий из FractionDimSize элементов
void Menu(Fraction **PFraction)
{
    char C = -1;
    int i, c;
    while (C != 'д')//д - выход
    {
        ShowMenu();
        std::cin >> C;
        while (C<'а' || C>'д')
        {
            std::cout << std::endl << "Неверно введен символ, введите еще раз: ";
            std::cin >> C;
        }
        switch (C)
        {
            case 'а'://Добавить объект
                if (Fraction::GetCount() == FractionDimSize)//Если массив заполнен
                {
                    std::cout << "Извините, массив заполнен" << std::endl;
                }
                else
                {
                    int C2 = -1;
                    while (C2 != 4)//4 - выход
                    {
                        ShowMenu2();
                        std::cin >> C2;
                        while (C2<1 || C2>4)
                        {
                            std::cout << std::endl << "Неверно введен номер, введите
еще раз: ";
                            std::cin >> C2;
                        }
                        int Index, IndexOfCopy;
                        switch (C2)
                        {
                            case 1://Значения по умолчанию
                                Index = Fraction::GetCount();
                                PFraction[Index] = new Fraction();
                                std::cout << "Количество объектов: " << Fraction::GetCount() << std::endl;
                                break;
                            case 2://Копия уже существующего в массиве объекта
                                if (Fraction::GetCount() == 0)
                                {
                                    std::cout << "Сначала добвьте объекты!";
                                    break;
                                }
                                std::cout << "Введите индекс копируемого объекта: ";
                                std::cin >> IndexOfCopy;
                                while (IndexOfCopy<0 || IndexOfCopy >= Fraction::GetCount())

```

```

        {
            std::cout << std::endl << "Индекс введен неверно,
введите еще раз: ";
            std::cin >> IndexOfCopy;
        }
        Index = Fraction::GetCount();
        PFraction[Index] = new Fraction(*PFraction[IndexOfCopy]);
        std::cout << "Количество объектов: " << Fraction::GetCount() << std::endl;
        break;
    case 3://Ввод значений числителя и знаменателя
        std::cout << "Числитель: ";
        int N,D;
        std::cin >> N;
        std::cout << "Знаменатель: ";
        std::cin >> D;
        Index = Fraction::GetCount();
        PFraction[Index] = new Fraction(N,D);
        std::cout << "Количество объектов: " << Fraction::GetCount() << std::endl;
        break;
    } //switch (C)
    } //while (C2 != 4) //4 - выход
    } //case 'a': //Добавить объект
    break;
case '6': //Удалить объект
    if (Fraction::GetCount() == 0)
    {
        std::cout << "Сначала добвьте объекты!";
        break;
    }
    std::cout << "Введите индекс удаляемого объекта: ";
    int Index;
    std::cin >> Index;
    while (Index < 0 || Index >= Fraction::GetCount())
    {
        std::cout << std::endl << "Индекс введен неверно, введите еще раз: ";
        std::cin >> Index;
    }
    c = Fraction::GetCount() - 1;
    for (i = Index; i < c; i++)
    {
        PFraction[i]->SetNumerator(PFraction[i+1]->GetNumerator());
        PFraction[i]->SetDenominator(PFraction[i + 1]->GetDenominator());
    }
    delete PFraction[i];
    std::cout << "Количество объектов: " << Fraction::GetCount() << std::endl;
    break;
case 'в': //Показать все объекты
    std::cout << "№" << "\t" << "Числитель" << "\t" << "Знаменатель" << std::endl;
    for (i = 0, c = Fraction::GetCount(); i < c; i++)
    {
        std::cout << i << "\t";
        PFraction[i]->Print();
    }
    break;
case 'г': //Управление объектом
    if (Fraction::GetCount() == 0)
    {
        std::cout << "Сначала добвьте объекты!";
        break;
    }
    std::cout << "Введите индекс управляемого объекта: ";
    std::cin >> Index;
    while (Index < 0 || Index >= Fraction::GetCount())
    {
        std::cout << std::endl << "Индекс введен неверно, введите еще раз: ";
        std::cin >> Index;
    }
}

```

```

int C3 = -1;
while (C3 != 5)//5- выход
{
    ShowMenu3();
    std::cin >> C3;
    while (C3 < 1 || C3>5)
    {
        std::cout << std::endl << "Неверно введен номер, введите еще раз: ";

        std::cin >> C3;
    }
    Fraction *F = new Fraction();//Создание объекта и инкремент Count;
    switch (C3)
    {
        case 1://Изменение содержимого объекта с заданным номером
            std::cout << "Новый числитель: ";
            int N, D;
            std::cin >> N;
            std::cout << "Новый знаменатель: ";
            std::cin >> D;
            PFraction[Index]->SetNumerator(N);
            PFraction[Index]->SetDenominator(D);
            break;
        case 2://Сокращение дроби объекта с заданным номером
            PFraction[Index]->Reduction();
            break;
        case 3://Умножение дроби объекта с заданным номером
            std::cout << "Числитель множителя: ";
            std::cin >> N;
            std::cout << "Знаменатель множителя: ";
            std::cin >> D;
            F->SetNumerator(N);
            F->SetDenominator(D);
            PFraction[Index]->Mul(*F);
            break;
        case 4://Деление дроби объекта с заданным номером
            std::cout << "Числитель делителя: ";
            std::cin >> N;
            std::cout << "Знаменатель делителя: ";
            std::cin >> D;
            F->SetNumerator(N);
            F->SetDenominator(D);
            PFraction[Index]->Div(*F);
            break;
    }
    delete F;//Удаление объекта и декремент Count
}
}
}

//Очистка динамической памяти
for (i = 0, c = Fraction::GetCount(); i < c; i++)
{
    delete PFraction[i];
}

}

"Evklid.cpp"
int Evklid(int a, int b)//Определение НОД по алгоритму Евклида
{
    int k;

    while (0 != (k = a%b))
    {
        a = b;
        b = k;
    }
    return (b>0)?b:-b;
}

```