Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

**Практическая работа №3**по дисциплине «Программирование на ЯВУ»  
на тему «Классы: Основные понятия и определения»

Выполнил:  
Студент Гоголин И.А.  
Группа И506Б  
  
Преподаватель:  
Скулябин М.А.

Санкт-Петербург  
2021 г.

# 1 Задача

Класс «Рациональное число» (РЧ). Поля: числитель, знаменатель. Методы: перегрузка операции приведения типа к float, операций умножения «\*» двух РЧ и РЧ с целым числом, сравнения на равенство «==» двух РЧ и РЧ с целым числом.

**Описание классов**

Класс RationalNumberClass:

1. Поля:
2. nominator типа double – хранит значение числителя рационального числа.
3. denominator типа double – хранит значение знаменателя рационального числа
4. Методы:
5. Конструктор по умолчанию, конструктор с параметрами (nominator и denominator), конструктор копирования.
6. Геттеры: getNominator – возвращает числитель, getDenominator – возвращае знаменталь; Сеттеры: setNominator – устанавливает значение числителя, setDenominator – устанааливает значение знаменателя.
7. Перегрузки:

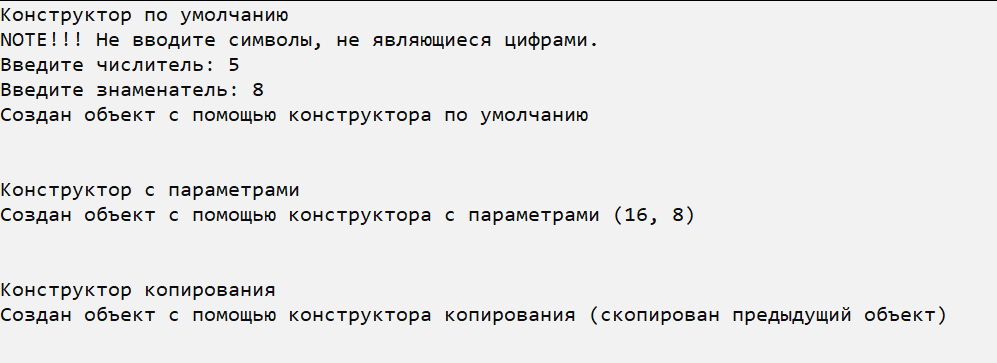
* Умножение РЧ на целое число (числитель РЧ умножается на целое число, переданное в параметрах, возвращается РЧ).
* Умножение РЧ на другое РЧ (числитель РЧ1 умножается на числитель РЧ2, знаменатель РЧ1 умножается на знаменатель РЧ2, возвращается РЧ1).
* Приведение РЧ к float (возвращает частное числителя и знаменателя РЧ, явно приведенное к float)
* Сравнение РЧ и целого числа (сравнивается частное числителя и знаменателя РЧ с числом, возвращается булево значение)
* Сравнение РЧ и другого РЧ (сравнивается частное числителя и знаменателя РЧ1 и частное и знаменатель РЧ2, возвращается булево значение)
* Ввод РЧ (реализуется посредством дружественных функций, вводится числитель и знаменатель)
* Вывод РЧ (реализуется посредством дружественных функций, выводится числитель и знаменатель)

1. Деструктор

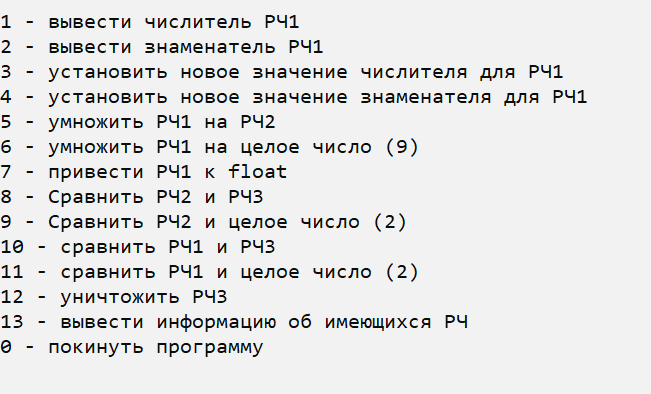
**Описание основной функции**

В функции int main() последовательно динамически выделяется память под объект класса RationalNumberClass, созданного с помощью конструктора по умолчанию, затем данные раннее созданного объекта мы вводим с клавиатуры. Далее динамически выделяем память под еще один объект, созданного с помощью конструктора с параметрами и по аналогии ещё один объект, но с помощью конструктора копирования. После создания трёх объектов будет открыто главное меню программы, где пользователь может протестировать все методы класса, описанные раннее.

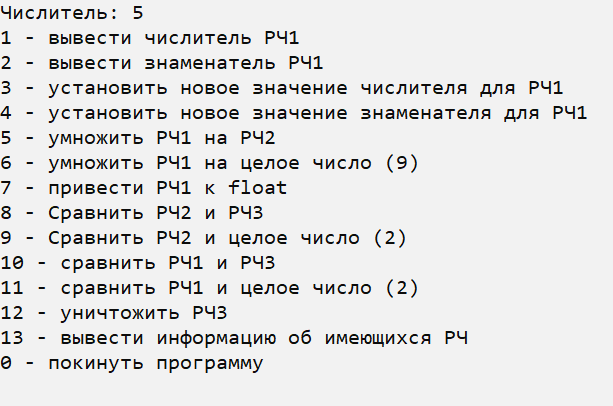
**Тестирование программы**



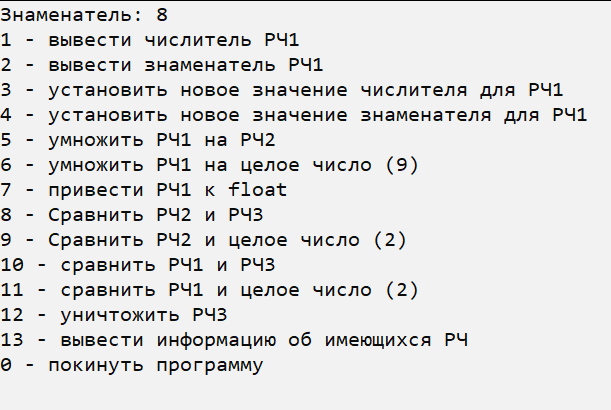
Создание объектов класса RationalNumberClass при запуске программы



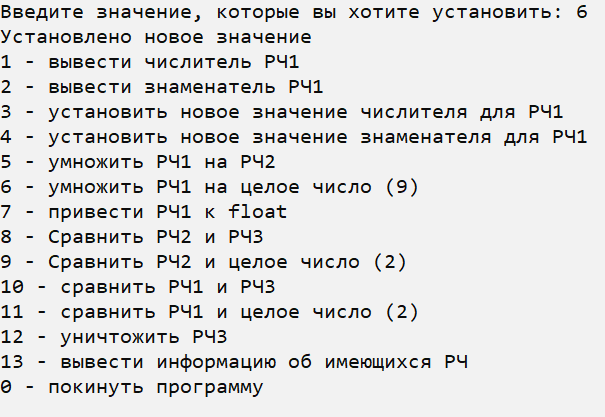
Главное меню программы



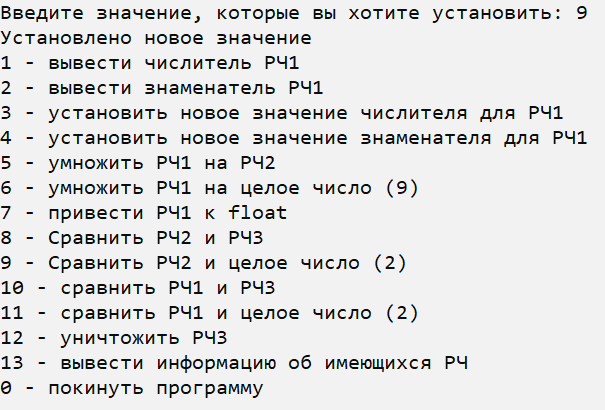
Вывод числителя с помощью геттера



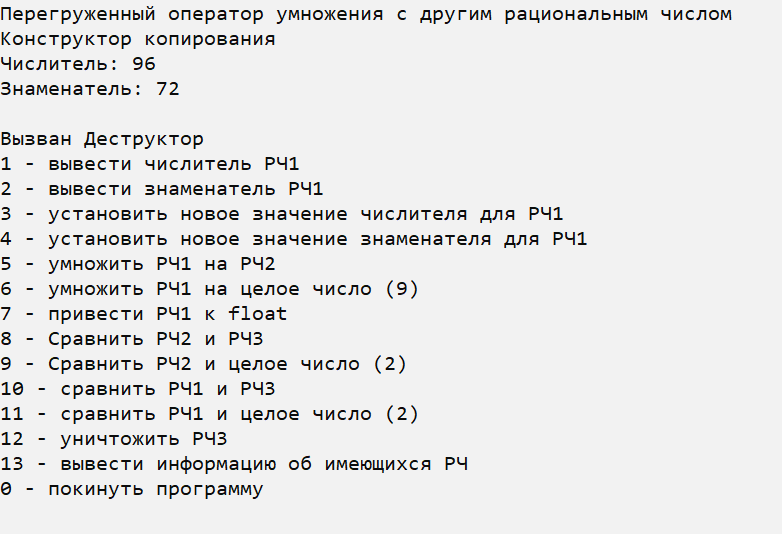
Вывод знаменателя с помощью геттера



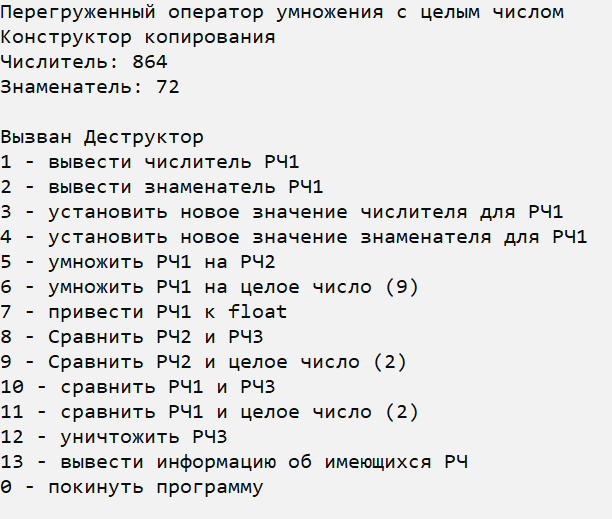
Новое значение числителя с помощью сеттера



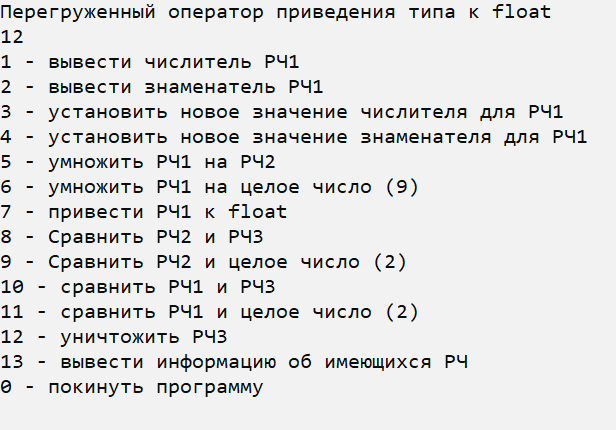
Новое значение знаменателя с помощью сеттера



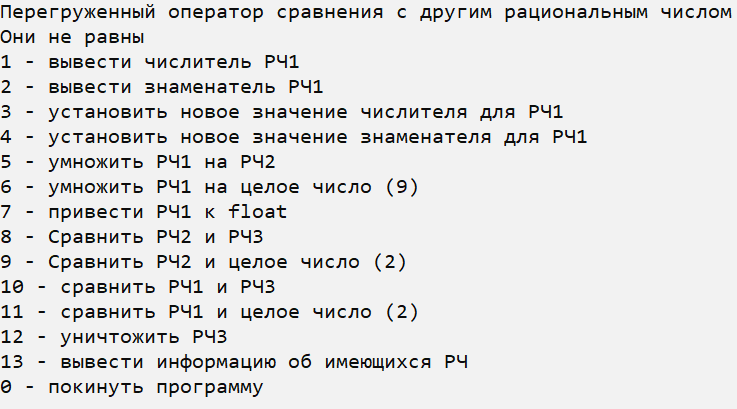
Умножение РЧ на другое РЧ



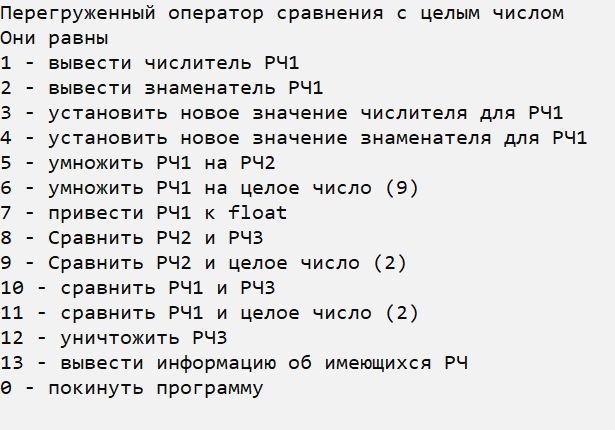
Умножение РЧ на целое число



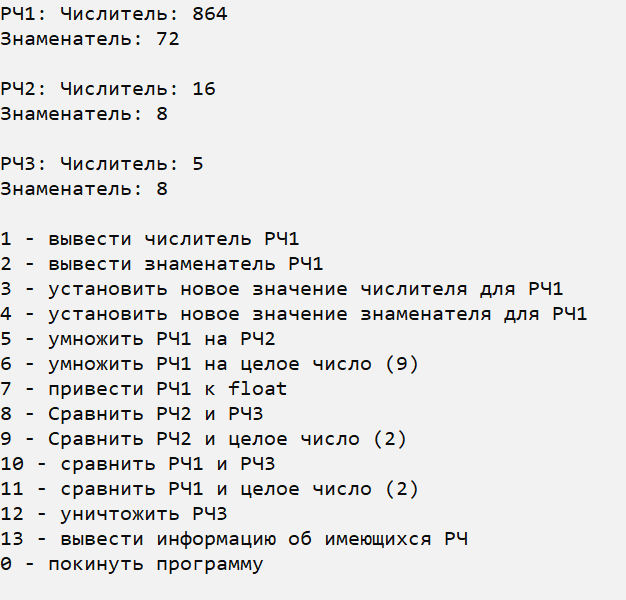
Приведение РЧ к float



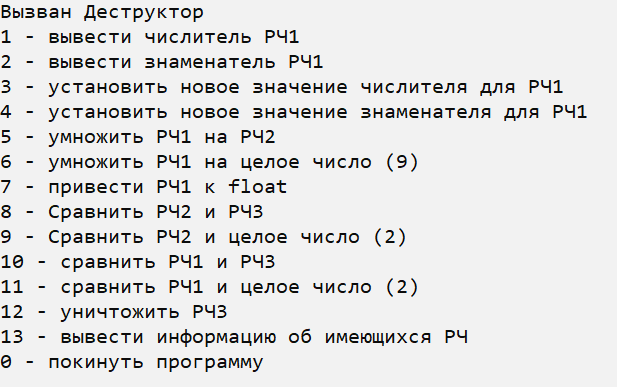
Сравнение РЧ с другим РЧ (пункт 8)



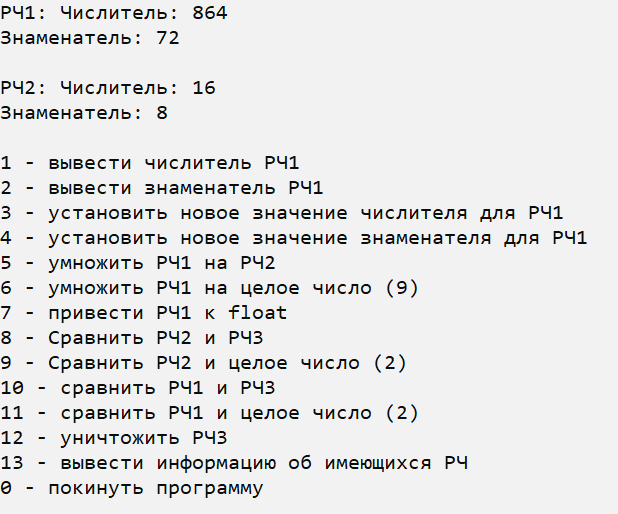
Сравнение РЧ с целым числом (пункт 9)



Вывод об имеющихся РЧ с помощью cout



Удалили РЧ3, освободили память



Снова вывели информацию обо всех имеющихся РЧ

**Текст программы**

#include <iostream>

#include <locale>

#include <cctype>

#include <cstdlib>

\*/

void cleaning() {

cin.clear();

cin.sync();

}

class RationalNumberClass {

private:

double nominator, denominator;

public:

//--------------------------------------------------------------------

RationalNumberClass() {

cout << "Конструктор по умолчанию" << endl;

};

RationalNumberClass(const double& u\_nominator, const double& u\_denominator) : nominator(u\_nominator), denominator(u\_denominator) {

cout << "Конструктор с параметрами" << endl;

};

RationalNumberClass(const RationalNumberClass& rc) : nominator(rc.nominator), denominator(rc.denominator) {

cout << "Конструктор копирования" << endl;

}

//--------------------------------------------------------------------

double getNominator() {

return nominator;

}

double getDenominator() {

return denominator;

}

void setNominator(int value) {

nominator = value;

cout << "Установлено новое значение" << endl;

}

void setDenominator(int value) {

denominator = value;

cout << "Установлено новое значение" << endl;

}

//--------------------------------------------------------------------

friend ostream& operator<< (ostream& out, const RationalNumberClass& rc);

friend istream& operator>> (istream& in, RationalNumberClass& rc);

//--------------------------------------------------------------------

RationalNumberClass operator \* (const RationalNumberClass& rc) {

nominator \*= rc.nominator;

denominator \*= rc.denominator;

cout << "Перегруженный оператор умножения с другим рациональным числом" << endl;

return \*this;

}

RationalNumberClass operator \* (const int& number) {

nominator \*= number;

cout << "Перегруженный оператор умножения с целым числом" << endl;

return \*this;

}

operator float() {

cout << "Перегруженный оператор приведения типа к float" << endl;

return ((float)(nominator / denominator));

}

bool operator== (const int& number) {

cout << "Перегруженный оператор сравнения с целым числом" << endl;

return ((float)(nominator / denominator) == number);

}

bool operator== (const RationalNumberClass& rc) {

cout << "Перегруженный оператор сравнения с другим рациональным числом" << endl;

return (nominator / denominator == rc.nominator / rc.denominator);

}

//--------------------------------------------------------------------

~RationalNumberClass() {

cout << "Вызван Деструктор" << endl;

}

};

ostream& operator<< (ostream& out, const RationalNumberClass& rc) {

out << "Числитель: " << rc.nominator << "\n" << "Знаменатель: " << rc.denominator << endl;

return out;

}

istream& operator>>(std::istream& in, RationalNumberClass& rc) {

int i = 0;

cout << "Введите числитель: ";

while (!(in >> rc.nominator) || in.get() != '\n') {

cout << "Разрешены только цифры!" << endl;

cleaning();

cout << "Введите числитель: ";

}

cout << "Введите знаменатель: ";

while (!(in >> rc.denominator) || in.get() != '\n') {

cout << "Разрешены только цифры!" << endl;

cleaning();

cout << "Введите знаменатель: ";

}

return in;

}

int main() {

int menuPos = 0;

int flag = 0;

int u\_nominator, u\_denominator;

system("color F0");

setlocale(LC\_ALL, "Russian");

RationalNumberClass\* RationalNumber1 = new RationalNumberClass;

cout << "NOTE!!! Не вводите символы, не являющиеся цифрами." << endl;

cin >> \*RationalNumber1;

cout << "Создан объект с помощью конструктора по умолчанию\n\n" << endl;

RationalNumberClass\* RationalNumber2 = new RationalNumberClass(16, 8);

cout << "Создан объект с помощью конструктора с параметрами (16, 8)\n\n" << endl;

RationalNumberClass\* RationalNumber3 = new RationalNumberClass(\*RationalNumber1);

cout << "Создан объект с помощью конструктора копирования (скопирован предыдущий объект)\n\n" << endl;

do {

cout

<< "1 - вывести числитель РЧ1\n"

<< "2 - вывести знаменатель РЧ1\n"

<< "3 - установить новое значение числителя для РЧ1\n"

<< "4 - установить новое значение знаменателя для РЧ1\n"

<< "5 - умножить РЧ1 на РЧ2\n"

<< "6 - умножить РЧ1 на целое число (9)\n"

<< "7 - привести РЧ1 к float\n"

<< "8 - Сравнить РЧ2 и РЧ3\n"

<< "9 - Сравнить РЧ2 и целое число (2)\n"

<< "10 - сравнить РЧ1 и РЧ3\n"

<< "11 - сравнить РЧ1 и целое число (2)\n"

<< "12 - уничтожить РЧ3\n"

<< "13 - вывести информацию об имеющихся РЧ\n"

<< "0 - покинуть программу\n\n\n"

<< "NOTE!!! Не вводите символы, не являющиеся цифрами."

<< endl;

cout << "Введите пункт меню: ";

while (!(cin >> menuPos) || cin.get() != '\n') {

cout << "Разрешены только цифры!" << endl;

cleaning();

cout << "Введите пункт меню: ";

}

system("CLS");

switch (menuPos) {

case 0: {

exit(0);

}

case 1: {

cout << "Числитель: " << RationalNumber1->getNominator() << endl;

break;

}

case 2: {

cout << "Знаменатель: " << RationalNumber1->getDenominator() << endl;

break;

}

case 3: {

int number;

cout << "Введите значение, которые вы хотите установить: ";

while (!(cin >> number) || cin.get() != '\n') {

cout << "Разрешены только цифры!" << endl;

cleaning();

cout << "Введите пункт меню: ";

}

RationalNumber1->setNominator(number);

break;

}

case 4: {

int number;

cout << "Введите значение, которые вы хотите установить: ";

while (!(cin >> number) || cin.get() != '\n') {

cout << "Разрешены только цифры!" << endl;

cleaning();

cout << "Введите пункт меню: ";

}

RationalNumber1->setDenominator(number);

break;

}

case 5: {

cout << (\*RationalNumber1 \* \*RationalNumber2) << endl;

break;

}

case 6: {

cout << (\*RationalNumber1 \* 9) << endl;

break;

}

case 7: {

cout << (float)\*RationalNumber1 << endl;

break;

}

case 8: {

if (\*RationalNumber2 == \*RationalNumber3) { cout << "Они равны" << endl; }

else { cout << "Они не равны" << endl; }

break;

}

case 9: {

if (\*RationalNumber2 == 2) { cout << "Они равны" << endl; }

else { cout << "Они не равны" << endl; }

break;

}

case 10: {

if (\*RationalNumber1 == \*RationalNumber3) { cout << "Они равны" << endl; }

else { cout << "Они не равны" << endl; }

break;

}

case 11: {

if (\*RationalNumber1 == 2) { cout << "Они равны" << endl; }

else { cout << "Они не равны" << endl; }

break;

}

case 12: {

delete RationalNumber3;

RationalNumber3 = 0;

break;

}

case 13: {

if (RationalNumber1 != NULL) {

cout << "РЧ1: " << \*RationalNumber1 << endl;

}

if (RationalNumber2 != NULL) {

cout << "РЧ2: " << \*RationalNumber2 << endl;

}

if (RationalNumber3 != NULL) {

cout << "РЧ3: " << \*RationalNumber3 << endl;

}

break;

}

default: {

cout << "Вы выбрали несуществующую команду, попробуйте снова." << endl;

break;

}

}

} while (flag == 0);

return 0;

}