УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационной безопасности

Кафедра инфокоммуникационных технологий

ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ
Часть 1

Лабораторная работа 5
Динамическая строка. Перегрузка операторов. База данных в Qt

Минск 2022

Макейчик Е.Г.

Содержание

Лабораторная работа 5
Динамическая строка. Перегрузка операторов.
База данных в Qt

Цель работы: Изучить работу с динамическими строками и перегрузку операторов. Научиться создавать базу данных в Qt.

Строки

Строки в C++ могут быть разделены на C-строки и C++-строки, где к первым относится использование одномерных массивов типа `char`, т.е. строка символов – это одномерный массив типа `char`, заканчивающийся нулевым байтом. Нулевой байт – это байт, каждый бит которого равен нулю, при этом для нулевого байта определена символьная константа '\0' (признак окончания строки или нуль-терминатор). Поэтому, если строка должна состоять из `k` символов, то в описании массива необходимо указать размер `k+1`, а при ручном формировании строки в ее окончание нужно явно добавить признак ее окончания.

Операции над строками выполняются только через стандартные функции:

- `strcpy_s(s1, s2)` – копирование содержимого `s2` в `s1`. Аргумент `s2` должен быть указателем на строку, оканчивающуюся нулем. Функция `strcpy()_s` возвращает указатель на `s1`.
- `strcat_s(s1, s2)` – присоединяет (конкатенирует) строку `s1` и копию строки `s2`. В конце модифицированной строки `s1` функция устанавливает нулевой символ. Нулевой символ, первоначально завершавший строку `s1`, замещается первым символом строки `s2`. Строка `s2` остается в первоначальном виде.

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "rus");
    char s1[] = "ООП – изи"; // Создание массива типа char. Размер массива не указан, массив будет равен размеру текста "ООП – изи" + 1 для нуль-терминатора '\0'
    char s2[40]; // Создание пустого массива типа char на 40 символов
    char s3[80]; // Создание пустого массива типа char на 80 символов
    strcpy_s(s2, "Я – программист"); /* Копирование в массив s2 строки "Я – программист". strcpy_s() – автоматически добавит нуль-терминатор '\0'.
                                    При попытке поместить в s2 строку большего размера чем она может вместить, будет выдана ошибка */
    strcpy_s(s3, s2); // Копирование в массив s3 строки хранящейся в s2.
    cout << "s1 = " << s1 << endl;
    cout << "s2 = " << s2 << endl;
    cout << "s3 = " << s3 << endl;
}
```

```
Консол...
s1 = ООП - изи
s2 = Я - программист
```

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "rus");
    char s1[] = "ООП – изи. "; // Создание массива типа char. Размер массива не указан, массив будет равен размеру текста "ООП – изи. " + 1 для нуль-терминатора '\0'
    char s2[40] = ""; // Создание пустого массива типа char на 40 символов
    char s3[80]; // Создание пустого массива типа char на 80 символов
    strcat_s(s2, s1); // Конкатенация содержимого строки s2 с содержимым строки s1. Результат будет сохранен в s2
    strcat_s(s2, "Я - программист"); // Конкатенация содержимого строки s2 со строкой "Я - программист". Результат будет сохранен в s2
    // strcat_s(s3, s2); Данная запись выдаст ошибку, так как массив s3 не содержит ничего, даже нуль-терминатора '\0'. Из-за этого функция не может определиться
    cout << "s1 = " << s1 << endl;
    cout << "s2 = " << s2 << endl;
}
```

```
Консоль отладки ...
s1 = ООП - изи.
s2 = ООП - изи. Я - программист
```

- `strcmp(s1, s2)` – сравнивает строку `s1` со строкой `s2` и возвращает значение 0, если строки равны, т.е. содержит одно и то же число одинаковых символов; значение 1, если s1>s2 и значение –1, если s1<s2.

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "rus");
    char s1[] = "ООП – изи. "; // Создание массива типа char. Размер массива не указан, массив будет равен размеру текста "ООП – изи. " + 1 для нуль-терминатора '\0'
    char s2[40] = "ООП – изи. "; // Создание массива типа char на 40 символов в котором будет храниться строка "ООП – изи. "
    char s3[80]= "Я - программист"; // Создание массива типа char на 80 символов в котором будет храниться строка "Я - программист"
    cout << "s1==s2 вернет: " << strcmp(s1, s2) << endl;
    cout << "s3>s2 вернет: " << strcmp(s3, s2) << endl;
    cout << "s2<s2 вернет: " << strcmp(s2, s3) << endl;
}
```

```
Консоль отла...
s1==s2 вернет: 0
s3>s2 вернет: 1
s2<s2 вернет: -1
```

- `strlen(s)` – возвращает длину строки, т.е. количество символов, начиная с нулевого и до нуль-терминатора, нулевой байт не учитывается.

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "rus");
    char s1[] = "ООП - изи. "; // Создание массива типа char. Размер массива не указан, массив будет равен размеру текста "ООП - изи. " + 1 для нуль-терминатора '\0'
    char s2[40] = "ООП - изи. "; // Создание массива типа char на 40 символов в котором будет храниться строка "ООП - изи. "
    char s3[80] = "Я программист"; // Создание массива типа char на 80 символов в котором будет храниться строка "Я - программист"
    cout << "Длина s1 = " << strlen(s1) << endl;
    cout << "Длина s2 = " << strlen(s2) << endl;
    cout << "Длина s3 = " << strlen(s3) << endl;
}
```

```
Конс...
Длина s1 = 11
Длина s2 = 11
Длина s3 = 13
```

- atoi(s) – преобразование строки s в число типа int. Строка должна содержать корректную запись целого числа. В противном случае возвращается 0.
- atol(s) – преобразование строки s в число типа long int. Строка должна содержать корректную запись длинного целого числа. В противном случае возвращается 0.
- atof(s) – преобразование строки s в число типа double. Строка должна содержать корректное число с плавающей точкой.

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "rus");
    char s1[] = "231"; // Создание массива типа char. Размер массива не указан, массив будет равен размеру текста "231" + 1 для нуль-терминатора '\0'
    char s2[40] = "42.23"; // Создание массива типа char на 40 символов в котором будет хранится строка "42.23"
    char s3[80] = "45"; // Создание массива типа char на 80 символов в котором будет хранится строка "45"
    int a = atoi(s1);
    int b = atoi(s3);
    double c = atof(s2);
    if (a > b)
    {
        cout << "s3>s1" << endl;
    }
    if (c > a)
    {
        cout << "s3>s2" << endl;
    }
    else
    {
        cout << "s2>s3" << endl;
    }
    if (atof(s2) < atof(s1))
    {
        cout << "s1>s2" << endl;
    }
}
```

```
s3>s1
s2>s3
s1>s2
```

- _itoa_s(int v, char s, int kod) – конвертирует целое число v в строчный эквивалент и помещает результат в строку s. Основание системы счисления для записи выходной строки определено параметром kod, который может принимать значения в интервале от 2 до 36.
- _ltoa_s(int v, char s, int kod) – конвертирует длинное целое число v в строчный эквивалент и помещает результат в строку s. Основание системы счисления для записи выходной строки определено параметром kod, который может принимать значения в интервале от 2 до 36.

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_CTYPE, "rus");
    int value; // Число
    char string[6] = ""; // Массив string типа char в котором будет храниться число
    cout << "Введите число: ";
    cin >> value;
    _itoa_s(value, string, 10); // Переводит value в десятичную систему и помещает в string
    cout << "Введенное число в десятичной системе: " << string << endl;
    _itoa_s(value, string, 16); // Переводим value в шестнадцатеричную систему и помещает в string
    cout << "Введенное число в шестнадцатеричной системе: " << string << endl;
    _itoa_s(value, string, 2); // Переводим value в двоичную систему и помещает в string
    cout << "Введенное число в двоичной системе: " << string << endl;
    system("pause");
    return 0;
}
```

```
C:\Users\Piglet\source\repos\ConsoleApp...
Введите число: 5
Введенное число в десятичной системе: 5
Введенное число в шестнадцатеричной системе: 5
Введенное число в двоичной системе: 101
```

Динамическая строка

Помимо традиционной декларации строки, можно создавать динамические строки.

```cpp
#include <iostream>
using namespace std;
int main()
{
    char* str;
    int size = 10;
    str = (char*)calloc(size, sizeof(char)); /* Выделение памяти указателю str для хранения 10 элементов.
    Не забываем, что в реальности последний элемент должен быть '\0', поэтому str в себе может хранить 9 символов.
    Учитывайте '\0' в конце, он всегда должен занимать последний элемент */
    strcpy_s(str, size, "123456789"); // Для заполнения str необходимо в функцию strcpy_s добавить еще один аргумент size (размер переменной str)
    cout << str;
    // strcpy_s(str, size, "1234567890"); Ошибка. Нехватка памяти для размещения строки "1234567890" в str (из-за того, что в конце должен стоять '\0' и символу '0' не хватает места)
}
```

```
123456789
```

Можно воспользоваться операторами new и delete для захвата и освобождения динамической памяти. Пример класса с конструктором для создания динамических строковых объектов и деструктором для их уничтожения после использования приведен в программе ниже.

```
#include <iostream>
using namespace std;
class Stroka
{
public:
    char* text; // Указатель на строку
    Stroka(const char* s) // Конструктор с параметром. В качестве параметра строка которую необходимо сохранить
    {
        text = new char[strlen(s) + 1]; /* Предполагается, что полученная строка s не будет иметь '\0' в конце,
                                           поэтому выделяется память для хранения строки s и добавляется + 1 для записи '\0' в конец */
        strcpy_s(text, strlen(s) + 1, s); // Помещение значения s в text
    }
    ~Stroka() // Деструктор
    {
        delete text; // Очистка памяти
    }
    void print() // Метод для вывода содержимого поля text
    {
        cout << text << endl;
    }
};
int main()
{
    Stroka t1("12345"), t2("6789"); // Создание объектов класса Stroka
    t1.print();
    t2.print();
}
```

```
12345
6789
```

В языке C++ для удобной работы со строками существует класс string. Для которого описанные выше операции и функции не требуются. Их можно реализовать при помощи операторов:

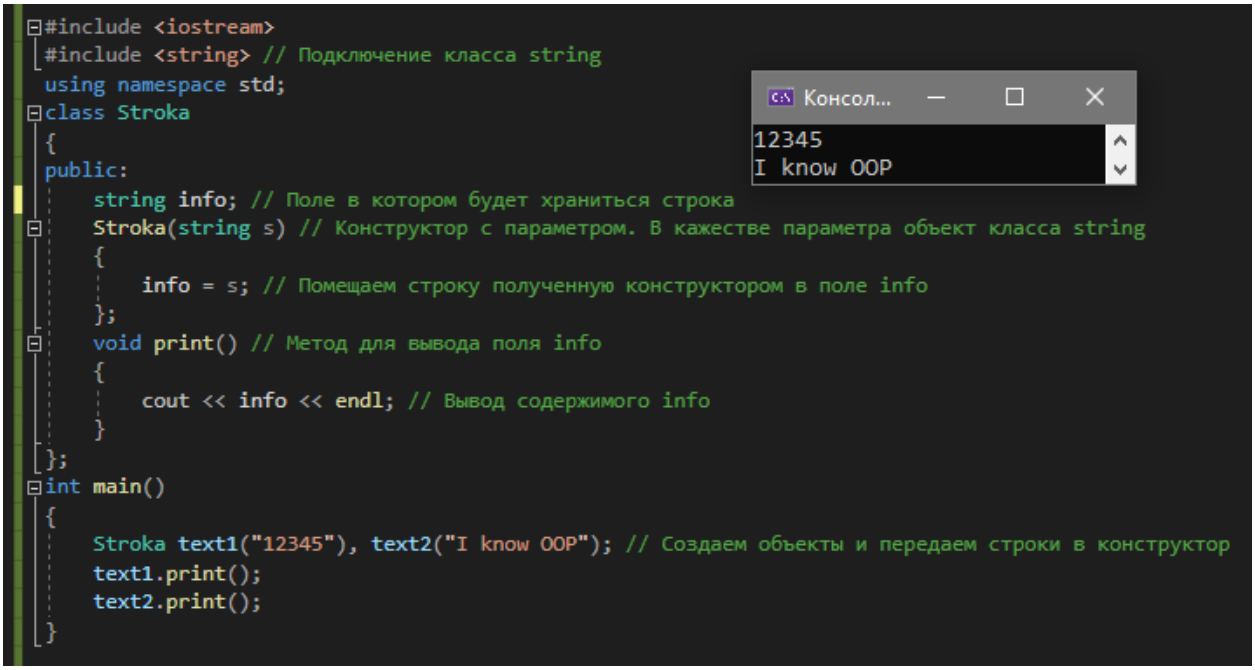
- = – присваивание.
- + – конкатенация (объединение строк).
- += – присваивание с конкатенацией.
- == – равенство.
- != – неравенство.
- < – меньше.
- <= – меньше или равно.
- >– больше.
- >= – больше или равно.
- [ ] – индексация.

```
#include <iostream>
#include <string> // Подключение класса string
using namespace std;
int main()
{
    string text1, text2 = "String - the Best. ", text3 = "I'm a programmer";
    text1 = text2; // Оператор "=" помещает значение text2 в text1
    cout << text1 << endl;
    text1 = text2 + text3; // Оператор "+" сцепляет два строковых объекта
    cout << text1 << endl;
    cout << "Length string: " << text1.length() << endl; // Метод length возвращает длинну строки
        if (text1 > text2) // Оператор ">" проверяет, что строковый объект слева от оператора больше строкового объекта справа от оператора
        {
            cout << "text1 > text2: " << (text1 > text2) << endl;
        }
        if (text1 == text1) // Оператор "==" проверяет, равен ли строковый объект слева от оператора строковому объекту справа от оператора
        {
            cout << "text1 == text1: " << (text1 == text1) << endl;
        }
        if (text2 > text1)
        {
        }
        else
        {
            cout << "text2 > text1: " << (text2 > text1) << endl;
        }
    int a;
    text1 = "123";
    a = stoi(text1); // Метод stoi преобразует последовательность символов в тип int
    cout << "Number = " << a << endl;
    text1 = to_string(7832.23); // Метод to_string преобразует значение в string
    cout << "Number = " << text1 << endl;
    text1 = to_string(a);
    cout << "Number = " << text1 << endl;
}
```

```
Выбрать Консоль отладки Microsoft Visual Studio
String - the Best.
String - the Best. I'm a programmer
Length string: 35
text1 > text2: 1
text1 == text1: 1
text2 > text1: 0
Number = 123
Number = 7832.230000
Number = 123
```

Объект класса string можно считать динамическим так как он сам выделяет памяти столько, сколько ему нужно для хранения информации.

В самом классе string уже прописан деструктор. Поэтому при его использовании в своем классе не нужно создавать деструктор для освобождения памяти.



Перегрузка операторов

В языке C++ наряду с обычными конструкциями операций над операндами со стандартными типами (арифметические, операции отношений, логические операции и т.д.) существует механизм перегрузки этих операторов, позволяющий манипулировать объектами классов пользователя, используя обычный синтаксис языка, т.е. привычную (удобную) запись операций. Если компилятор обнаружил, что в программе есть перегрузка операторов для объектов конкретного типа, введенных пользователем, то, найдя эти операции, компилятор анализирует их. И, если операнды имеют встроенный тип данных, то будет заложена стандартная операция, а если операнды имеют тип данных пользователя, т.е. являются объектами класса, для которого операция была перегружена, то будет заложена перегруженная операция, запрограммированная пользователем.

Операторы для конкретного класса перегружаются или методом этого класса, или дружественной функцией этого класса:

```
имя_класса operator@(параметры)
{
код, определяющий смысл указанной операции
}
```

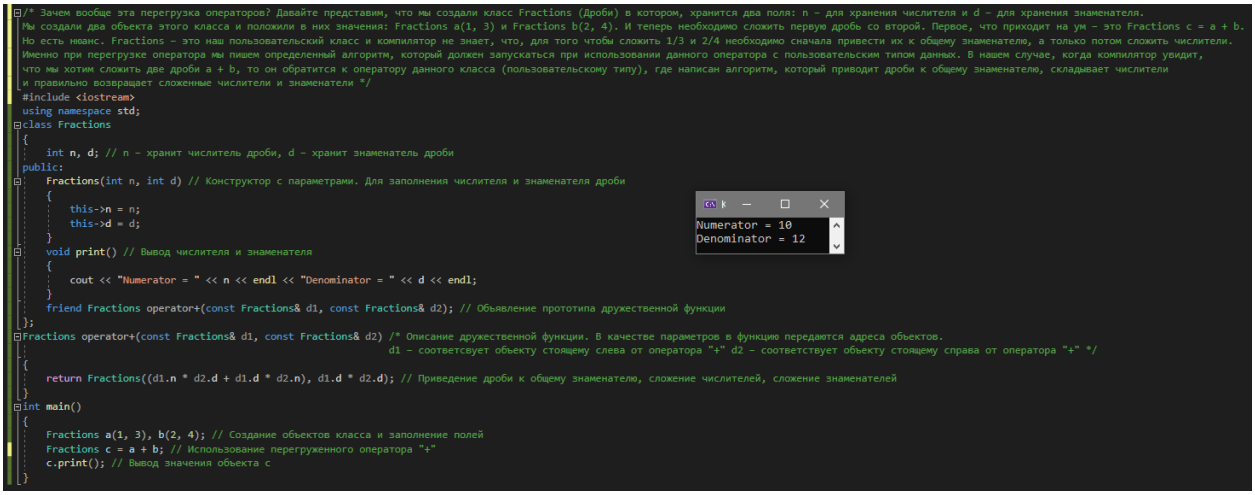
где @ – символ перегружаемого оператора.

Для перегрузки унарных и бинарных операторов действуют следующие требования:

- унарный оператор может быть перегружен методом класса без параметров или дружественной функцией с одним параметром, т.к. у метода имеется первый скрытый параметр (указатель this), который автоматически устанавливается на единственный операнд унарной операции, а дружественная функция такого указателя не имеет.

- бинарный оператор перегружается методом класса с одним параметром или дружественной функцией с двумя параметрами, т.к. при его перегрузке методом на первый операнд устанавливается скрытый указатель this, а второй передается через параметр; при перегрузке дружественной функцией первый операнд передается через первый параметр, второй – через второй параметр.

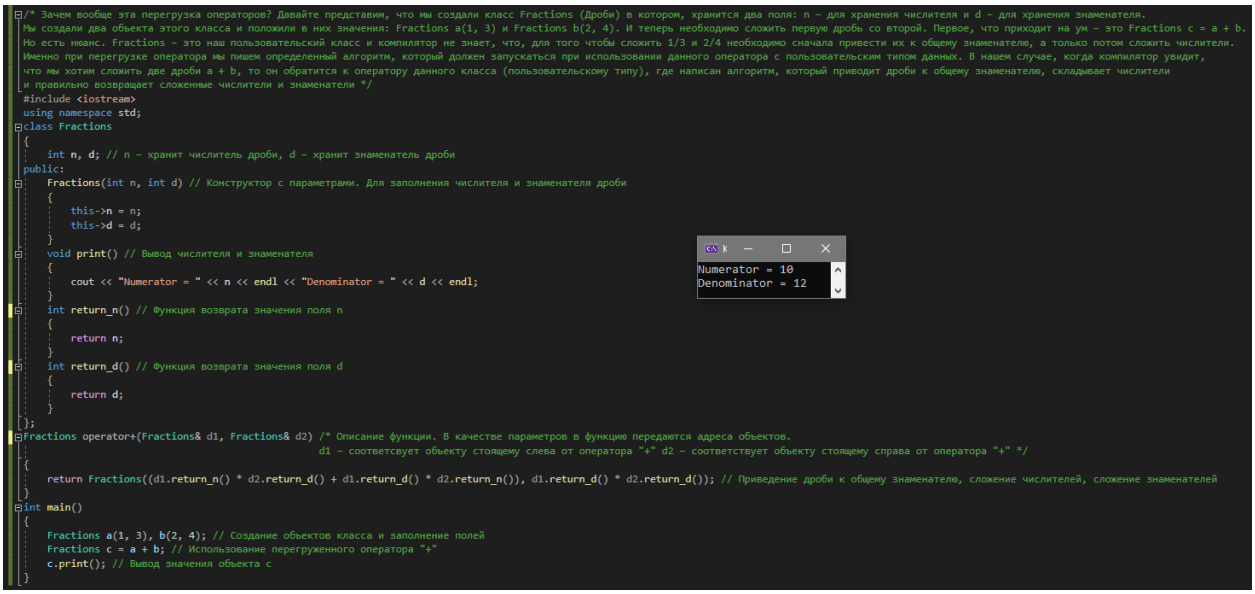
В программе ниже продемонстрирован пример перегрузки оператора через дружественную функцию.




```
/* Зачем вообще эта перегрузка операторов? Давайте представим, что мы создали класс Fractions (Дроби) в котором, хранится два поля: n – для хранения числителя и d – для хранения знаменателя.
Мы создали два объекта этого класса и положили в них значения: Fractions a(1, 3) и Fractions b(2, 4). И теперь необходимо сложить первую дробь со второй. Первое, что приходит на ум – это Fractions c = a + b.
Но есть нюанс. Fractions – это наш пользовательский класс и компилятор не знает, что, для того чтобы сложить 1/3 и 2/4 необходимо сначала привести их к общему знаменателю, а только потом сложить числители.
Именно при перегрузке оператора мы пишем определенный алгоритм, который должен запускаться при использовании данного оператора с пользовательским типом данных. В нашем случае, когда компилятор увидит,
что мы хотим сложить две дроби a + b, то он обратится к оператору данного класса (пользовательскому типу), где написан алгоритм, который приводит дроби к общему знаменателю, складывает числители
и правильно возвращает сложенные числители и знаменатели */
#include <iostream>
using namespace std;
class Fractions
{
    int n, d; // n – хранит числитель дроби, d – хранит знаменатель дроби
public:
    Fractions(int n, int d) // Конструктор с параметрами. Для заполнения числителя и знаменателя дроби
    {
        this->n = n;
        this->d = d;
    }
    void print() // Вывод числителя и знаменателя
    {
        cout << "Numerator = " << n << endl << "Denominator = " << d << endl;
    }
    friend Fractions operator+(const Fractions& d1, const Fractions& d2); // Объявление прототипа дружественной функции
};
Fractions operator+(const Fractions& d1, const Fractions& d2) /* Описание дружественной функции. В качестве параметров в функцию передаются адреса объектов.
                                                              d1 – соответсвует объекту стоящему слева от оператора "+" d2 – соответствует объекту стоящему справа от оператора "+" */
{
    return Fractions((d1.n * d2.d + d1.d * d2.n), d1.d * d2.d); // Приведение дроби к общему знаменателю, сложение числителей, сложение знаменателей
}
int main()
{
    Fractions a(1, 3), b(2, 4); // Создание объектов класса и заполнение полей
    Fractions c = a + b; // Использование перегруженного оператора "+"
    c.print(); // Вывод значения объекта c
}
```

Операторы можно перегружать, используя и обычные функции. Единственное в чем они уступают дружественным функциям это, тем, что они не имеют доступ к полям и методам с модификатором доступа private. Поэтому перегрузки оператора через обычные функции используют только если этому оператору не нужны значения полей с модификатором доступа private или в классе предусмотрены методы, которые смогут вернуть значение полей с доступом private.

В программе ниже продемонстрирован пример перегрузки оператора через обычную функцию.




```
/* Зачем вообще эта перегрузка операторов? Давайте представим, что мы создали класс Fractions (Дроби) в котором, хранится два поля: n – для хранения числителя и d – для хранения знаменателя.
Мы создали два объекта этого класса и положили в них значения: Fractions a(1, 3) и Fractions b(2, 4). И теперь необходимо сложить первую дробь со второй. Первое, что приходит на ум – это Fractions c = a + b.
Но есть нюанс. Fractions – это наш пользовательский класс и компилятор не знает, что, для того чтобы сложить 1/3 и 2/4 необходимо сначала привести их к общему знаменателю, а только потом сложить числители.
Именно при перегрузке оператора мы пишем определенный алгоритм, который должен запускаться при использовании данного оператора с пользовательским типом данных. В нашем случае, когда компилятор увидит,
что мы хотим сложить две дроби a + b, то он обратится к оператору данного класса (пользовательскому типу), где написан алгоритм, который приводит дроби к общему знаменателю, складывает числители
и правильно возвращает сложенные числители и знаменатели */
#include <iostream>
using namespace std;
class Fractions
{
    int n, d; // n – хранит числитель дроби, d – хранит знаменатель дроби
public:
    Fractions(int n, int d) // Конструктор с параметрами. Для заполнения числителя и знаменателя дроби
    {
        this->n = n;
        this->d = d;
    }
    void print() // Вывод числителя и знаменателя
    {
        cout << "Numerator = " << n << endl << "Denominator = " << d << endl;
    }
    int return_n() // Функция возврата значения поля n
    {
        return n;
    }
    int return_d() // Функция возврата значения поля d
    {
        return d;
    }
};
Fractions operator+(Fractions& d1, Fractions& d2) /* Описание функции. В качестве параметров в функцию передаются адреса объектов.
                                                  d1 – соответсвует объекту стоящему слева от оператора "+" d2 – соответствует объекту стоящему справа от оператора "+" */
{
    return Fractions((d1.return_n() * d2.return_d() + d1.return_d() * d2.return_n()), d1.return_d() * d2.return_d()); // Приведение дроби к общему знаменателю, сложение числителей, сложение знаменателей
}
int main()
{
    Fractions a(1, 3), b(2, 4); // Создание объектов класса и заполнение полей
    Fractions c = a + b; // Использование перегруженного оператора "+"
    c.print(); // Вывод значения объекта c
}
```

Результат у получится таким же, что и при использовании дружественной функции. Единственное отличие – это способ получения значения полей n и d.

Перегрузка операторов через методы класса очень похожа на перегрузку операторов через дружественные функции. Но, при перегрузке оператора через метод класса, доступ к объекту, для которого вызван оператор можно получить при помощи указателя this.

```cpp
/* Зачем вообще эта перегрузка операторов? Давайте представим, что мы создали класс Fractions (Дроби) в котором, хранится два поля: n – для хранения числителя и d – для хранения знаменателя.
Мы создали два объекта этого класса и положили в них значения: Fractions a(1, 3) и Fractions b(2, 4). И теперь необходимо сложить первую дробь со второй. Первое, что приходит на ум – это Fractions c = a + b.
Но есть нюанс. Fractions – это наш пользовательский класс и компилятор не знает, что, для того чтобы сложить 1/3 и 2/4 необходимо сначала привести их к общему знаменателю, а только потом сложить числители.
Именно при перегрузке оператора мы пишем определенный алгоритм, который должен запускаться при использовании данного оператора с пользовательским типом данных. В нашем случае, когда компилятор увидит,
что мы хотим сложить две дроби a + b, то он обратится к оператору данного класса (пользовательскому типу), где написан алгоритм, который приводит дроби к общему знаменателю, складывает числители
и правильно возвращает сложенные числители и знаменатели */
#include <iostream>
using namespace std;
class Fractions
{
    int n, d; // n – хранит числитель дроби, d – хранит знаменатель дроби
public:
    Fractions(int n, int d) // Конструктор с параметрами. Для заполнения числителя и знаменателя дроби
    {
        this->n = n;
        this->d = d;
    }
    void print() // Вывод числителя и знаменателя
    {
        cout << "Numerator = " << n << endl << "Denominator = " << d << endl;
    }
    Fractions operator+(Fractions& d2) // Метод для перегрузки оператора "+"
    {
        return Fractions((this->n * d2.d + this->d * d2.n), this->d * d2.d); // this – указывает на объект который стоит слева от оператора "+"
    };
    Fractions operator+(int d2) // Метод для перегрузки оператора "+"
    {
        return Fractions(this->n + (d2 * this->d), this->d); // Возврат дроби с новыми значениями
    };
};

int main()
{
    Fractions a(1, 3), b(2, 4); // Создание объектов класса и заполнение полей
    Fractions c = a + b; // Использование перегруженного оператора "+"
    c.print(); // Вывод значения объекта c
}
```

```
Консоль о...
Numerator = 10
Denominator = 12
```

В программах ниже рассмотрены примеры решения задачи на работу со строками с использованием перегрузки оператора. Задача решена разными способами: с использованием массива char и класса string. В задаче было необходимо ввести строку символов s1 (признак окончания ввода строки – нажатие клавиши Enter). Использовать перегруженный оператор "=", для копирования строки s1 в s2 с удалением из нее двух первых и двух последних символов.

```cpp
// Решение задачи с использованием массива char
#include <iostream>
using namespace std;
class Stroka
{
public:
    char text[256]; // Создание массива char
    void print() // Метод для вывода содержимого поля text
    {
        cout << text << endl;
    }
    void operator=(Stroka& str) // Перегрузка оператора "=". str – объект стоящий справа от "="
    {
        for (int i = 2; i < strlen(str.text) - 2; i++) // Заполнение массива char объекта стоящего слева от "="
            {
            text[i - 2] = str.text[i];
            }
        text[strlen(str.text) - 4] = '\0'; // Добавление '\0' для обозначение конца строки
    }
};
int main()
{
    Stroka s1, s2; // Создание объектов
    gets_s(s1.text); // Помещение строки, введенной пользователем в поле text объекта s1
    s1.print(); // Вывод строки
    s2 = s1; // Запись в поле text объекта s2 строки из объекта s1 без двух первых и последних символов, при помощи перегрузки оператора "="
    s2.print(); // Вывод строки
}
```

```
Ко...
programming
programming
ogrammi
```

```cpp
// Решение задачи с использованием класса string
#include <iostream>
#include <string>
using namespace std;
class Stroka
{
public:
    string text; // Создание объекта text класса string
    void print() // Метод для вывода содержимого поля text
    {
        cout << text << endl;
    }
    void operator=(Stroka& str) // Перегрузка оператора "=". str – объект стоящий справа от "="
    {
        for (int i = 2; i < str.text.length() - 2; i++) // Прохождение по строке в объекте str. Обход начинается сразу с третьего символа, не доходя до конца строки на два символа
            {
                text += str.text[i]; // Добавление в строку text i-го символа из строки объекта str
            }
    }
};
int main()
{
    Stroka s1, s2; // Создание объектов
    getline(cin, s1.text); // Помещение строки введенной пользователем в поле text объекта s1
    cin.clear(); // Очищение потока для ввода
    s1.print(); // Вывод строки
    s2 = s1; // Запись в поле text объекта s2 строки из объекта s1 без двух первых и последних символов, при помощи перегрузки оператора "="
    s2.print(); // Вывод строки
}
```

```
Ко...
programming
programming
ogrammi
```

База данных в Qt

Для создания и работы с БД используются классы QSqlDatabase, QSqlQuery, QSqlTableModel (документация). Создадим базу данных и реализуем возможности добавлять и удалять строки БД, смотреть.

mainwindows.h

```h
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QSqlDatabase> // Подключение QSqlDatabase
#include <QDebug> // Класс для отладки программ
#include <QSqlQuery> // Класс для запросов
#include <QSqlTableModel> // Класс для таблиц

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_tableView_clicked(const QModelIndex &index);

private:
    Ui::MainWindow *ui;
    QSqlDatabase db; // Создание объекта db класса QSqlDatabase
    QSqlQuery *query; // Создание указателя на обект query класса QSqlQuery
    QSqlTableModel *model; // Создание указателя на обект model класса QSqlTableModel


    int row; // Номер активной строки
};
#endif // MAINWINDOW_H
```

mainwindows.cpp

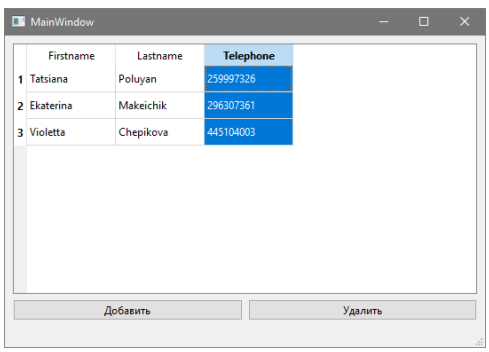
```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    // Настройка подключения к БД
    db = QSqlDatabase::addDatabase("QSQLITE"); // В метод addDatabase передаем драйвер QSQLITE
    db.setDatabaseName("./test.db"); // Устанавливаем путь и имя файла. Если прописаны ./, то файл создается в папке с программой
    // Если БД откроется, вывести в консоль "open", иначе "no open"
    if(db.open())
    {
        qDebug("open");
    }
    else
    {
        qDebug("no open");
    }
    query = new QSqlQuery(db); // Инициализация запроса с использованием конструктора
    query->exec("CREATE TABLE TelephoneBook(Firstname TEXT, Lastname TEXT, Telephone BIGINT);"); // Запрос. Таблица с указанными полями

    model = new QSqlTableModel(this, db); // Инициализация таблицы с использованием конструктора
    model->setTable("TelephoneBook"); // В метод setTable передаем таблицу TelephoneBook
    model->select(); // Вызов метода select

    ui->tableView->setModel(model); // Устанавливаем model в tableView
}

MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::on_pushButton_clicked()
{
    model->insertRow(model->rowCount()); // При нажатии на кнопку будет добавляться строка в конце таблицы
}
void MainWindow::on_pushButton_2_clicked()
{
    model->removeRow(row); // Передаем переменную row в метод
}
void MainWindow::on_tableView_clicked(const QModelIndex &index) // Метод работает с индексом ячейки на которую происходит нажатие
{
    row=index.row(); // Берем из ячейки номер строки и помещаем его в переменную row
}
```



Задание к лабораторной работе 5

Для выполнения лабораторной работы необходимо установить и настроить Visual Studio и инструменты Qt. Задание к лабораторной работе 5 состоит из нескольких задач. Задачи выполняются по вариантам (например, Вариант 1 – номера по списку в группе: 1, 9, 17, 25; Вариант 2 – номера по списку в группе: 2, 10, 18, 26 и т.д.).

Вариант 1

Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

Вас попросили разобраться с читерами в популярной MMORPG игре"Sword, Magic, Shield, Bow, Arrow and something else". Сервер постоянно возвращает какие-то непонятные строки типа: "qqfdsdvn3u9r8329fjf239 n392hfnvowgn we 3wrfiodg" и т.д. (никто не понимает, что означают эти строки, возвращаемые сервером игры, но если работает – то не трогай). Но одно разработчики знают точно, если в строке, возвращаемой сервером, встречаются символы в верхнем регистре к примеру: "fdghjHJHKLdsjf FG F ddsgsdgwfg3 dsggsege", то это значит, что кто-то использует читы в игре. Вы опытный разработчик и знаете, как выполнять такие задачи. Если символы в верхнем регистре – это читеры, то сервер просто должен перестать показывать их. Полные энтузиазма вы приступили к работе. Вам необходимо создать класс Server_Answer в котором будет поле для хранения строки, метод для просмотра этого поля и перегруженный оператор "=", который скопирует строку, введенную пользователем в поле, предназначенное для хранения строки в классе Server_Answer удалив оттуда все символы в верхнем регистре.

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения логов необходима база данных. Вам необходимо создать БД (поля: исходная строка, обработанная строка) с возможностью добавления и удаления строк.

Вариант 2

Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

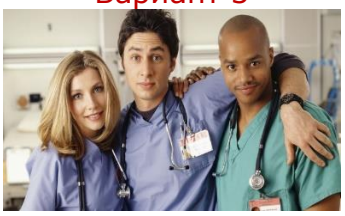
Ваша помощь агрономам была неоценима, но сбор урожая закончился и вас перевели в лабораторию математического моделирования. В это лаборатории работают умнейшие люди, в голове которых одни числа из-за чего общаться они тоже начали на языке цифр, а точнее вместо символа они используют его код из ASCII Table. К примеру, фраза "Hello" у них будет записана вот так "72 101 108 108 111". Вы пока не выучили ASCII Table и не можете так быстро переводить код, поэтому решили написать программу, которая в этом вам поможет. Вам необходимо создать класс Mathematical_Language в котором будет поле для хранения строки, метод для просмотра этого поля, и перегруженный оператор "=", который поместит строку, введенную пользователем в поле предназначенное для хранения строки, заменив все символы введенной строки на ее код в ASCII Table (у пробела тоже есть свой код).

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации о строках необходима база данных. Вам необходимо создать БД (поля: исходная строка, код строки) с возможностью добавления и удаления строк.

Вариант 3



Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

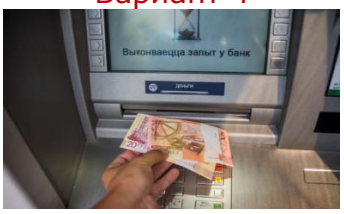
Вчера вы прочитали прекрасный журнал "IT-сфера". Если нам лень что-то делать, мы напишем программу, которая сделает это за нас" в котором была интересная статья о том, что можно отсканировать лист с рукописным текстом и он перевести этот рукописный текст в редактируемые форматы текста с которой сможет работать компьютер. Вам сразу пришла идея о том, что такая программа не помешала бы в аптечных киосках больницы. Пришел пациент, приложил рецепт к сканеру, и фармацевт уже несет нужное лекарство. Однако программе будет сложно понять, где закончился диагноз и началось название лекарства. Поэтому вы дали указание врачам писать названия лекарств в фигурных скобках "{ }". Текст в этих скобках и будет являться названием лекарства. Осталось мелочь, создать программу, которая будет выводить все эти названия, заключенные между фигурными скобками. Вам необходимо создать класс Recipe в котором будет поле для хранения строки, метод для просмотра этого поля, и выполнить перегрузку оператора "=", который выберет из строки введенной пользователем подстроки заключенные между фигурными скобками "{ }" и поместит их в поле предназначенное для хранения (фигурных скобок может быть несколько).

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации о лекарствах необходима база данных. Вам необходимо создать БД (поля: исходная строка, обработанная строка) с возможностью добавления и удаления строк.

Вариант 4



Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

Вашему начальнику постоянно приходит куча счетов. И одним прекрасным днем ему это все надоело, и он попросил вас написать программу, которая из этой кучи текста выберет для него только числа. Задача ясна, и вы приступили к работе. Вам необходимо создать класс Bills_For_Communal в котором будет поле для хранения строки, метод для просмотра этого поля, и выполнить перегрузку оператора "=", который выберет все числа из строки введенной пользователем и поместит их в поле предназначенное для хранения (числа будут только положительными и целыми, т.е есть после обработки строки "afsfefe77-3 9.2 dg3 -24" должно вернуть "77 3 9 2 3 24").

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации о счетах необходима база данных. Вам необходимо создать БД (поля: исходная строка, обработанная строка) с возможностью добавления и удаления строк.

Вариант 5

Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

13 сентября ваш полицейский участок был переполнен задержанными. Программистов было так много, что вы решили дать им самим заполнить документы. Спустя время, вы увидели, что документы заполнены какими-то числами типа "72 101 108 108 111 32 102 114 111 109 32 116 104 101 32 97 117 116 104 111 114". Посидев немного и вспомнив с кем, вы имели дело, вас осенило, что эти символы являются кодами для ASCII Table. Пожелав им побольше пропущенных ";" в коде, вы приступили к написанию программы, которая переведет эти числа в нормальные символы. Вам необходимо создать класс Report в котором будет поле для хранения строки, метод для просмотра этого поля, и выполнить перегрузку оператора "=", который переведет числа из введенной пользователем строки в символы из ASCII Table.

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации необходима база данных. Вам необходимо создать БД (поля: исходная строка, символы) с возможностью добавления и удаления строк.

Вариант 6

Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

Вы потихоньку наводите порядок в своем городе и теперь решили издать пару важных распоряжений. Вот только прочитав еще раз свои распоряжения, вас посетило чувство, что чего-то не хватает. Через пару минут стало понятно, что вас смутило. Чтобы показать важность распоряжений, необходимо чтобы все буквы в нем были в верхнем регистре. Вы решили написать программу, которая сделает это быстрее чем Ctrl+A, Shift+F3. Вам необходимо создать класс Document в котором будет поле для хранения строки, метод для просмотра этого поля, и выполнить перегрузку оператора "=", который переведет все символы в верхней регистр (в строке используются только буквы латинского алфавита).

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации необходима база данных. Вам необходимо создать БД (поля: исходная строка, обработанная строка) с возможностью добавления и удаления строк.

Вариант 7

Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

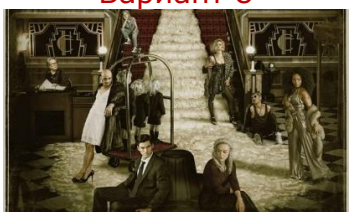
Прочитав книгу "Криптография для взрослых" вы резко захотели заняться этим интересным делом. Вы поставили у себя в офисе новый, мощный компьютер с процессором Core i9-12900KS и решили написать программу, которая бы шифровала текст путем замены символа в вашей строке символом, который стоит следующим в ASCII Table. К примеру, при вводе "I am Batman", получим "J.bn.Cbunbo". Вам необходимо создать класс Encrypt в котором будет поле для хранения строки, метод для просмотра этого поля, и выполнить перегрузку оператора "=", который заменит символы исходной строки на символы, стоящие следующими в ASCII Table.

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации необходима база данных. Вам необходимо создать БД (поля: исходная строка, обработанная строка) с возможностью добавления и удаления строк.

Вариант 8



Задача 1

Задача 1 выполняется с использованием Visual Studio. Пользовательский класс должен содержать поле для хранения строки (поле может быть типа char или string на ваше усмотрение). Реализовать метод вывода содержимого этого поля, хранящего строку и реализовать перегрузку оператора "=" (любым из трех способов, рассмотренных выше).

Сегодня в отеле очень скучно и вы придумали себе интересное задание. Вы прошлись по отелю у просили у каждого встречного написать какую ни будь строку, а после определяли является ли эта строка палиндромом. Но строки были такими длинными и их было так много, что вы решили написать программу, которая проверит все эти строки и выведет вам только строки палиндромы. Вам необходимо создать класс Palindrome в котором будет поле для хранения строки, метод для просмотра этого поля, и выполнить перегрузку оператора "=", который выберет слова палиндромы из введенной строки (слова в строке разделяются одним или несколькими пробелами, все символы кроме пробела считаются частью слова) К примеру при вводе строки "saippuakivikauppias 123 22 3,3 yuy ;7; ..rr ?" должно выводиться "saippuakivikauppias 22 3,3 yuy ;7; ?".

Задача 2

Задача 2 выполняется с использованием инструментов Qt. Необходимо дать поясняющие комментарии к коду.

Для хранения информации необходима база данных. Вам необходимо создать БД (поля: исходная строка, обработанная строка) с возможностью добавления и удаления строк.