Практическое занятие №4

«Перегрузка операций»

(Продолжительность работы 2 часа)

Цели:

Изучить механизм перегрузки операций.

1. Краткие теоретические сведения

Перегрузка операций – это одна из самых мощных и полезных возможностей ООП. Её применение позволяет существенно упростить написание программ, сделать их понятнее за счет применения интуитивно понятных обозначений, таких как +, *, [ ] и т.д.

Перегрузка операций, по сути, дает возможность переопределить стандартный язык программирования. Используя классы для создания новых типов переменных и перегрузку для определения операций над этими типами, можно, по сути, создать новый, собственный язык программирования, заточенный под конкретную задачу, и, таким образом, перейти от общего к предметно-ориентированному программированию.

Благодаря перегрузке операций любой класс можно сделать таким, что он будет вести себя подобно встроенному типу данных. В классе можно перегрузить любые из следующих операций:

+	—	*	/	%	^	&	|	~	!	=
<	>	+=	-=	*=	/=	%=	^=	&=	|=	<<
>>	>>=	<<=	= =	! =	<=	>=	&&	||	++	-
			( )	[ ]	new	delete		new[ ]	delete []	

Нельзя перегружать операции

. -> .* :: ?:

Функции-операции, реализующие перегрузку операций, имеют вид

operator операция ([операнды]) ;

Если функция является элементом класса, то первый операнд соответствующей операции будет самим объектом, для которого вызвана функция-операция. В случае одноместной операции список параметров будет пуст. Для двухместных операций функция будет иметь один параметр, соответствующий второму операнду. Если функция-операция не является элементом класса, она будет иметь один параметр в случае одноместной операции и два — в случае двухместной.

Для перегрузки операций существуют такие правила:

- Приоритет и правила ассоциации для перегруженных операций остаются теми же самыми, что и для операций над встроенными типами.
- Нельзя изменить поведение операции по отношению к встроенному типу.
- Функция-операция должна быть либо элементом класса, либо иметь один или несколько параметров типа класса.
- Функция-операция не может иметь аргументов по умолчанию.

В качестве примера приведем перегрузку операции сложения и умножения (как скалярного произведения) для класса, описывающего вектор в пространстве с координатами x, y, z.

```
class my_vector {// Объявляем класс - вектор длины 3
        public:    // с элементами-данными x, y и z - координатами вектора
        long double x;
        long double y;
        long double z;
        //Переопределяем оператор умножения
        long double operator*(my_vector v2);
        //Переопределяем оператор сложения
        my_vector operator+(my_vector v2);
};
// переопределяем оператор умножения * как
// скалярное произведение векторов
long double my_vector::operator*(my_vector v2)
{       long double help;
        help=x*v2.x + y*v2.y + z*v2.z;
        return help;
 };

//переопределяем оператор сложения + как сумму векторов
my_vector my_vector::operator+(my_vector v2)
{       my_vector help;
        help.x = x + v2.x;
        help.y = y + v2.y;
        help.z = z + v2.z;
        return help;
};
```

После этого в программе для объектов класса my_vector будут доступны операции сложения и умножения:

```
        my_vector v1, v2, v3;
        long double a;
        v1 = v2 + v3;
        a = v1 * v2;
```

При перегрузке операций необходимо помнить следующее:

- C++ не умеет образовывать из простых операций более сложные. Например, в классе со сложением строк мы определили присваивание и сложение; но это не значит, что тем самым будет автоматически определено присвоение суммы (+=). Такую операцию нужно реализовывать отдельно.
- Невозможно изменить синтаксис перегруженных операций. Одноместные операции должны быть одноместными, а двухместные — двухместными.
- Нельзя изобретать новые обозначения операций. Возможные операции ограничиваются тем списком, что приведен в начале этого раздела.
- Желательно сохранять смысл перегружаемой операции. Например, конкатенация — естественная семантика сложения для строк.

Операции не обязательно объявлять членами класса. Скажем, предыдущий пример с перегрузкой операций сложения и умножения для класса *my_vector* можно реализовать и так:

```
class my_vector {  // Объявляем класс - вектор длины 3
        public:     // с элементами-данными x, y и z - координатами вектора
        long double x;
        long double y;
        long double z;
        };
//переопределяем оператор умножения * как
//скалярное произведение векторов
long double operator*( my_vector v1, my_vector v2)
{       long double help;
        help=v1.x*v2.x + v1.y*v2.y +v1. z*v2.z;
        return help;
};

//переопределяем оператор сложения + как сумму векторов
my_vector operator+( my_vector v1, my_vector v2)
{       my_vector help;
        help.x = v1.x + v2.x;
        help.y = v1.y + v2.y;
        help.z = v1.z + v2.z;
        return help;
};
```

Несколько слов о перегрузке унарных операций.

Имеется особенность синтаксиса при перегрузке операций с префиксной и постфиксной формой ++ (инкремент) и -- (декремент).

В случае перегрузки префиксной формы используют следующий синтаксис переопределения:

```
void operator++( );
```

Если требуется переопределить постфиксную форму, то прототип перегружаемой операции будет таким:

```
void operator++( int );
```

Различие состоит лишь в том, что у постфиксной формы в скобках стоит *int*. Здесь int не играет роль аргумента и не означает целое число. Это просто сигнал для компилятора, чтобы использовалась постфиксная версия оператора.

Имеется также особенность перегрузки операции [ ] – индексация массива. Так как данная операция часто используется не только для доступа на чтение, но и для доступа на запись (то есть что-то типа *x[i] = y*), то целесообразно сделать так, чтобы перегруженная функция operator [] возвращала свое значение по ссылке. Это будет выглядеть примерно следующим образом:

```
int& operator[](int i);
```

2. Практическое задание (100%)

Все классы следует наделить конструкторами, деструктором. Необходимо явно реализовать конструктор копирования и перегрузить оператор присваивания. Необходимо подготовить демонстрацию по работе перегруженных для класса операторов.

Варианты:

1. Создать класс ПРЯМОУГОЛЬНИК со сторонами параллельными осям координат (прямоугольная система координат OXY). Реализовать метод вывода на экран информации о прямоугольнике. Перегрузить бинарный оператор несимметрической разности двух прямоугольников (-); унарный оператор (-): симметричное отображение прямоугольника относительно оси координат OX и OУ. Следует учесть, что результатом выполнения оператора может быть пустой прямоугольник.

2. Создать класс, описывающий тип ВРЕМЯ. Класс должен включать в себя атрибуты, описывающие часы, минуты, секунды и миллисекунды и иметь метод для вывода времени на экран. Для данного класса перегрузить следующие бинарные операторы: суммы(+), разности (-).

3. Создать класс МНОГОЧЛЕН степени ***n*** от одной переменной ***x***, задаваемый массивом своих коэффициентов (массив должен храниться в динамической памяти и задаваться внутри конструктора, используя датчик случайных чисел). Класс должен включать конструктор, которому в качестве параметра передается степень многочлена; деструктор; конструктор копирования, метод, который печатает уравнение на экран. Для данного класса перегрузить следующие бинарные операторы: суммы двух многочленов (+), разности двух многочленов (-), операцию присваивания (=).

4. Для пространства $\mathbf{R}^3$ (выбрана правая система декартовых прямоугольных координат {0, **i, j, k**}) создать класс ВЕКТОР, предусмотрев для него несколько видов конструкторов, метод для вывода на экран его координат. Для данного класса перегрузить следующие бинарные операторы: суммы(+), разности (-),“векторное произведение” (*).

5. Создать класс ОТРЕЗОК ЧИСЛОВОЙ ПРЯМОЙ (сегмент). Предусмотреть несколько конструкторов для создания объектов класса, реализовать метод вывода отрезка на экран. Перегрузить следующие бинарные операторы: дополнение одного сегмента другим (+), пересечение двух сегментов (*), несимметрическая разность сегментов (-) {[1,5]-[3,6]=[1,3]}. Следует учесть, что результатом выполнения оператора может быть пустой сегмент, в этом случае следует вернуть отрезок нулевой длины [0,0].

6. Создать класс КВАДРАТНАЯ МАТРИЦА 3X3. Элементы матрицы следует хранить в динамической памяти и задавать внутри конструктора, используя датчик случайных чисел. Класс должен включать конструктор; деструктор; конструктор копирования; метод для вывода матрицы на экран. Перегрузить следующие бинарные операторы: сумма матриц (+), произведение матриц (*).

7. Создать класс ПРЯМОУГОЛЬНИК со сторонами параллельными осям координат (прямоугольная система координат OXY). Реализовать метод вывода на экран информации о прямоугольнике. Перегрузить следующие бинарные операторы: пересечение двух прямоугольников (*), объединение двух прямоугольников (+). Следует учесть, что результатом выполнения оператора может быть пустой прямоугольник.

8. Создать класс ОКРУЖНОСТЬ в прямоугольной системе координат OXY. Реализовать метод вывода на экран информации об окружности. Перегрузить бинарный оператор несимметрической разности двух окружностей (-); унарный оператор (-):

симметричное отображение окружности относительно оси координат OX и OY. Следует учесть, что результатом выполнения оператора может быть окружность нулевого радиуса.

9. *Создать класс СТРОКА. Объект класса должен характеризоваться следующими свойствами: длина строки, динамически выделяемый массив символов, заканчивающийся символом ‘\n’, для хранения элементов строки. Предусмотреть несколько конструкторов для создания объектов класса, в том числе и конструктор с параметром, который задает длину будущей строки. Для данного класса реализовать метод вывода строки на экран, перегрузить следующие бинарные операторы: сцепление строк (+), удаление подстроки из строки (-) {“qwert”-“we”=qrt”, {“qwerty”-“tu”=“qwerty”, “qwert”-“qwert”=”” }. Учесть тот случай, когда результатом операции может быть пустая строка (строка нулевой длины).

10. Создать класс СЕКТОР ЕДИНИЧНОГО КРУГА в прямоугольной системе координат OXY с центром в точке O(0,0), который определяется двумя различными точкам на окружности единичного радиуса. Реализовать несколько конструкторов, метод вывода на экран информации о секторе. Перегрузить бинарные операторы: пересечение двух секторов (*), объединение двух секторов (+), несимметрическая разность двух секторов (-). Следует учесть все специальные случаи.

11. *Создать класс МНОГОЧЛЕН степени ***n*** от одной переменной ***x***, задаваемый массивом своих коэффициентов (массив должен храниться в динамической памяти и задаваться внутри конструктора, используя датчик случайных чисел). Класс должен включать конструктор, которому в качестве параметра передается степень многочлена; деструктор; конструктор копирования, метод, который печатает уравнение на экран. Для данного класса перегрузить бинарный оператор произведения двух многочленов (*), унарный оператор (-).

12. *Создать класс ОКРУЖНОСТЬ в прямоугольной системе координат OXY. Реализовать метод вывода на экран информации об окружности. Перегрузить следующие бинарные операторы: пересечение двух окружностей (*), объединение двух окружностей (+). Следует учесть, что результатом выполнения оператора может быть окружность нулевого радиуса.

13. Создать класс КОЛЬЦО (геометрическая фигура на плоскости) в прямоугольной системе координат OXY с центром в точке O(0,0), которая определяется двумя радиусами (R1, R2, R2>R1>0). Реализовать метод вывода на экран информации о кольце. Перегрузить бинарные операторы: пересечение двух колец (*), объединение двух колец (+), несимметрическая разность двух колец (-). Следует учесть, что результатом выполнения оператора может быть получено кольцо с равными радиусами (R1=R2), в этом случае следует сообщить об ошибке.

14. Создать класс ПРЯМОУГОЛЬНИК СО СТОРОНОЙ НА ОСИ OX, для прямоугольной системы координат OXY (одна сторона которого расположена на оси OX). Реализовать несколько конструкторов, метод вывода на экран информации о прямоугольнике. Перегрузить бинарные операторы: пересечение двух прямоугольников (*), объединение двух прямоугольников (+), несимметрическая разность двух прямоугольников (-). Следует учесть все специальные случаи.

15. Создать класс КВАДРАТНАЯ МАТРИЦА 3X3. Элементы матрицы следует хранить в динамической памяти и задавать внутри конструктора, используя датчик случайных чисел. Класс должен включать конструктор; деструктор; конструктор копирования; метод для вывода матрицы на экран. Перегрузить бинарный оператор

разности двух матриц, операцию присваивания, унарный оператор (-), который возвращает транспонированную матрицу.

3. Список рекомендуемой литературы

1. Объектно-Ориентированное программирование: учебник / Г. С. Иванова, Т. Н. Ничушкина ; под общ. ред. Г. С. Ивановой. — М. : Изд-во МГТУ им. Н. Э. Баумана, 2014. — 455.

2. Professional C++, 3rd Edition. Marc Gregoire. ISBN: 978-1-118-85805-9. Paperback 984 pages. September 2014

4. Контрольные вопросы

1. Для чего в C++ применяется перегрузка операций
2. Истинно ли следующее утверждение: операция >= может быть перегружена?
3. Сколько аргументов требуется для определения перегруженной унарной операции?
4. Сколько аргументов требуется для определения перегруженной бинарной операции?
5. Чем отличается действие операции ++ в префиксной форме от её действия в постфиксной форме?
6. Истинно ли следующее утверждение: перегруженная операция всегда требует на один аргумент меньше, чем количество операндов?
7. Когда перегружается операция арифметического присваивания, то результат
    a. Передается объекту справа от операции
    b. Передается объекту слева от операции
    c. Передается объекту, вызвавшему операцию
    d. Должен быть возвращен
8. Истинно ли следующее утверждение: компилятор не выдаст сообщение об ошибке, если вы перегрузите операцию * для выполнения деления?
9. Существуют ли операции, которые нельзя перегружать?