Лабораторная работа №4

по теме: "Перегрузка функций. Перегрузка операторов"

Перегрузка функций

Перегрузка функций – это использование одного имени для нескольких функций. Смысл перегрузки в том, что каждое переопределение функции должно использовать либо другие типы параметров, либо другое их количество.

Например,

```
#include <iostream>
using namespace;

int myfunc (int i); // Эти функции различаются типами параметров
double  myfunc (double i);

int main()
{
cout<<myfunc(10)<<"     "; // Вызов функции myfunc (int i)
cout<<myfunc(5.5)<<"    "; // Вызов функции myfunc (double i)

return 0;
}

double myfunc (double i)
{
return i;
}

int  myfunc (int i)
{
return i;
}
```

Задание

1. Проработайте пример приведенный в лабораторной работе. Объясните результаты.

2. Составьте программу для решения задачи. Найдите периметр треугольника, заданного координатами своих вершин (считать, что треугольник существует). Разработайте функции нахождения расстояния между двумя точками, заданными своими координатами. Предусмотрите только случаи двумерного и трехмерного пространств.

Перегрузка операторов

Перегрузка операторов напоминает перегрузку функций и является одним из видов перегрузки функций, но при этом перегружаемый оператор всегда связан с классом. Например, в классе, поддерживающем стек, оператор "+" можно перегрузить для добавления элемента в стек, а оператор "-" для выталкивания элементов из стека. Перегружаемый оператор сохраняет свое первоначальное значение, просто набор типов, к которым его можно отнести расширяется. После перегрузки операции над объектами новых классов выглядят точно так же, как операции над встроенными типами. Кроме того, перегрузка операторов лежит в основе системы ввода-вывода в языке C++.

Перегрузка операторов производится с помощью *операторных функции*, которые определяют действия перегружаемых операторов применительно к соответствующему кассу. Операторные функции создаются с помощью ключевого слова **operator**. Операторные функции могут быть как членами класса, так и обычными функциями. Как правило, обычные операторные функции объявляются дружественными по отношению к классу, для которого они перегружают оператор.

Создание операторной функции-члена имеет следующий вид:

тип_возвращаемого_значения имя_класса::operator#(список аргументов)
{ . . . *// Операции* }

Обычно операторная функции возвращает объект класса, с которым она работает, тем не менее, тип возвращаемого значения может быть любым. Символ # заменяется перегружаемым оператором. Например, если в классе перегружается оператор умножения "*" операторная функция-член называется ***operator****. При перегрузке унарного оператора список аргументов остается пустым. При перегрузке бинарного оператора список аргументов содержит один параметр.

Пример. Программа создает класс *Комплексное число*, в котором хранятся реальная и мнимая части числа и перегружается операция сложения *"+"* .

```
#include <iostream.h>

class Complex {
        double Re,Im;
public:
        Complex() {}
        Complex(double a, double b){ Re=a; Im=b;}
        void show() {
        cout<<"("<<Re<<","<<Im<<")\n";
        }
        Complex operator+(Complex ob);
        Complex operator++();
};

// перегрузка "+" для класса Complex
Complex Complex::operator+(Complex ob)
{
Complex temp;
temp.Re=ob.Re+Re;
temp.Im=ob.Im+Im;
return temp;
}

// перегрузка префиксного инкремента "++" для класса Complex
Complex Complex::operator++()
{
Re++;
Im++;
return *this; //возврат объекта, генерирующего вызов
}


int main() {
```

```
Complex ob1(10,5), ob2(7,11);
ob1.show(); //вывод на экран (10,5)
ob2.show(); //вывод на экран (7,11)

ob1=ob1+ob2;
ob1.show();  //вывод на экран (17,16)

ob2++;
ob2.show();  //вывод на экран (8,12)

return 0;
}
```

Функция **operator+** имеет только один параметр, а перегружает бинарный оператор (двухместную операцию). Причина заключается в том, что операнд, стоящий в левой части оператора, передается операторной функции неявно с помощью указателя this. Операнд, стоящий в правой части оператора, передается через параметр ob.

Вывод: при перегрузке бинарного оператора вызов операторной функции генерируется объектом, стоящим в левой части оператора.

Допускается следующее выражение: (ob1+ob2).show(); //вывод на экран суммы ob1+ob2 В этой ситуации операторная функция создает новый объект, который уничтожается после возвращения из функции show().

Замечание. При вызове функции-члена ей неявно передается указатель на вызывающий объект. Этот указатель называется this. Указатель this автоматически передается всем функциям-членам. Дружественные функции не являются функциями-членами, им не передается указатель this, статические функции-члены также не получают этот указатель.

Задание

1.Определите класс date (дата), содержащий три закрытых члена типа int: day (день), month (месяц), year (год) и массив, определяющий количество дней каждого месяца: days[13]={0,31,28,31,30,31,30,31,31,30,31,30,31}. Напишите конструкторы класса (Сколько конструкторов необходимо?) и функцию, показывающую дату. Перегрузите бинарные операции "+" и "-", которые выполняют следующие действия: "даты + дата", "дата – дата", изменение даты на заданное число дней: "дата + int", "int+дата" (две последние операции различны, перестановка операторов транслятором не производится), "дата-int", унарные операции "++" и "--" (переход к следующей дате, к предыдущей дате). В функции main() покажите работы определенных в классе перегружаемых операций.

1. Для работы строительной фирмы определите класс Date (дата), содержащий три закрытых члена типа
int: day (день), month (месяц), year (год) и массив, определяющий количество дней каждого месяца: days[13]={0,31,28,31,30,31,30,31,31,30,31,30,31}.
2. Напишите конструкторы класса (Сколько конструкторов необходимо?).
3. Перегрузите операцию вставки в поток (<<), выводящую дату в поток вывода.
4. Перегрузить операцию "даты + дата", которая будет определять срок сдачи объекта, зная дату начала и период времени строительства объекта. Например,

дата начала строительства: 01.04.2010
период строительства: 2 года 3 месяца 10 дней
дата сдачи объекта: 11.07.2012

Так же перегрузите обратную операцию «дата» – «дата», определяющую срок начала строительства, если известны дата сдачи объекта и период строительства.

5. Для определения даты поставки строительных материалов необходимо перегрузить операцию вычисления даты через определенное количество дней. Для этого перегрузите операции «дата» + «int», «int» + «дата» (две последние операции различны, перестановка операторов транслятором не производится).
6. Для оформления заработной платы рабочим необходимо знать дату, с которой начисляется заработная плата. Если известно количество отработанных дней и дата начисления зарплаты, то для определения исходной даты может быть выполнена операция «вычитания». Перегрузите операцию: «дата» – «int» → «дата».
7. Для оформления отчетных документов необходимо перегрузить унарные операции "++" и "--" для перехода к следующей или к предыдущей дате.
8. В функции main() покажите работы определенных в классе перегружаемых операций.