Лабораторная работа №3.
Перегрузка операций

Цель работы: изучение возможностей перегрузки операций в Си++.

Содержание

Теоретические сведения

1.1 Общие сведения

Кроме перегрузки функций C++ позволяет организовать перегрузку операций. Механизм перегрузки операций позволяет обеспечить более традиционную и удобную запись действий над объектами. Для перегрузки встроенных операторов используется ключевое слова ***operator***.

Синтаксически перегрузка операций осуществляется следующим образом:

```
тип operator @ (список_параметров-операндов)
{
  // ... тело функции ...
}
```

где:

@ — знак перегружаемой операции (-, +, * и т. д.),

тип — тип возвращаемого значения.

Тип возвращаемого значения должен быть отличным от ***void***, если необходимо использовать перегруженную операцию внутри другого выражения.

Например, функция перемножения матрицы и вектора может быть записана следующим образом:

```
class matrix;  // прототип класса matrix
class vect {
  int *p;
  int size;
  public:
    ...
  friend vect operator *(const vect &, const matrix &);
};

class matrix {
  int **base;
  int column_size, row_size;
  public:
    ...
  friend vect operator *(const vect &, const matrix &);
};
vect operator *(const vect &v, const matrix &m)
{...}


int main() {
  vect v(3);
  matrix m(2,3);
  v.in();
  m.in();
```

```
  vect r= v * m; // функция умножения
  r.out();
  cin.get();
  cin.get();
  return 0;
}
```

Любой перегруженный оператор можно вызвать с использованием функциональной формы записи (функции-операции):

```
  r = operator *(v, m);
```

Функция-операция описывается и может вызываться так же, как любая другая функция. Использование операции – это лишь сокращенная запись явного вызова функции операции.

Пример:

```
void f(complex a, complex b) {
  complex c = a + b;            // сокращенная запись
  complex d = operator+(a,b);   // явный вызов
}
```

Имеется два способа описания функции, соответствующей переопределяемой операции:

- если функция задается как обычная функция-элемент класса, то первым операндом операции является объект класса, указатель на который передается неявным параметром ***this;***
- если первый операнд переопределяемой операции не является объектом некоторого класса, либо требуется передавать в качестве операнда не указатель, а сам объект (значение), то соответствующая функция должна быть определена как дружественная классу с полным списком аргументов.

Можно описывать функции, определяющие значения следующих операций:

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

Операции, не допускающие перегрузки:

- . - прямой выбор члена объекта класса;
- .* - обращение к члену через указатель на него;
- ?: - условная операция;
- :: - операция указания области видимости;
- sizeof - операция вычисления размера в байтах;
- # - препроцессорная операция.

1.2 Правила перегрузки операций

- Язык C++ не допускает определения для операций нового лексического символа, кроме уже определенных в языке. Например, нельзя определить в качестве знака операции @.
- Не допускается перегрузка операций для встроенных типов данных. Нельзя, например, переопределить операцию сложения целых чисел:

```
int operator +(int i, int j);
```

- Нельзя переопределить приоритет операции.
- Нельзя изменить синтаксис операции в выражении. Например, если некоторая операция определена как унарная, то ее нельзя определить как бинарную. Если для операции используется префиксная форма записи, то ее нельзя переопределить в постфиксную. Например, ***!a*** нельзя переопределить как ***a!***
- Перегружать можно только операции, для которых хотя бы один аргумент представляет тип данных, определенный пользователем. Функция-операция должна быть определена либо как функция-член класса, либо как внешняя функция, но дружественная классу.

Функция- член класса	**Дружественная функция**

`class String {`
`  ...`
`  public: String operator + (const String &);`
`  ...`
`};`
	`class String {`
	`  ...`
	`  public: friend String operator +(String &, String &);`
	`  ...`
	`};`

1.3 Перегрузка унарной операции

Если унарная операция перегружается как функция-член, то она не должна иметь аргументов, так как в этом случае ей передается неявный аргумент-указатель ***this*** на текущий объект.

Если унарная операция перегружается дружественной функцией, то она должна иметь один аргумент – объект, для которого она выполняется.

Таким образом, для любой унарной операции **@**, **aa@** или ***@aa*** может интерпретироваться или как ***aa.operator@()***, или как ***operator@(aa)***. Если определена и та, и другая, то и **aa@** и **@aa** являются ошибками.

Функция- член класса	Дружественная функция

`class A`
`{ ...`
`  public: A operator !();`
`  ...n`
`};`
	`class A`
	`{`
	`  ...`
	`  public: friend A operator !(A);`
	`...`
	`};`

1.4 Перегрузка бинарной операции

Если бинарная операция перегружается с использованием функции-члена, то в качестве своего первого аргумента она получает неявно переданную переменную класса (указатель ***this*** на объект), а в качестве второго — аргумент из списка параметров. То есть, фактически, бинарная операция, перегружаемая функцией-членом, имеет один аргумент (правый операнд), а левый передается неявночерез указатель ***this***.

```
class complex {
  double real;
  double imag;
  public:
    complex operator +(const complex &);
    ...
};

complex complex :: operator +(complex &c) {
  complex temp;
  temp.real=this->real+c.real;
  temp.imag=this->imag+c.imag;
  return(temp);
}
```

Если бинарная операция перегружается дружественной функцией, то в списке параметров она должна иметь оба аргумента:

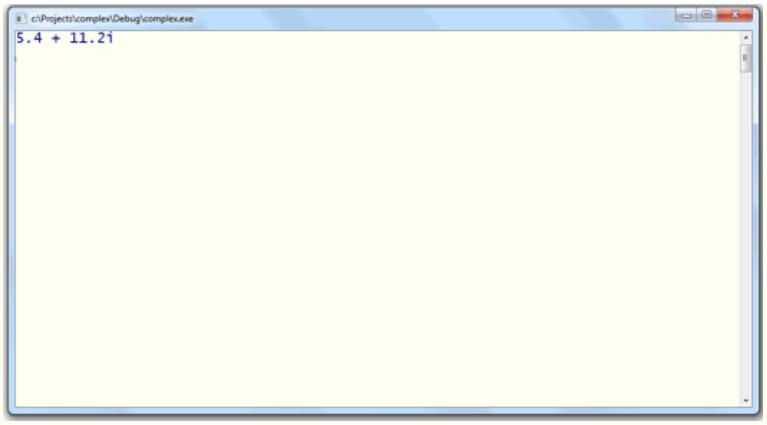
```
#include <iostream.h>
  using namespace std;
  class complex {
    double real;
    double imag;
    public: complex(double r=0, double i=0){
          real = r;
          imag = i;
        }
        void out(void){
                cout << real << " + " << imag << "i" << endl;
        }

        friend complex operator + (const complex &c1, const complex &c2);
  };

  complex operator + (const complex & c1, const complex &c2) {
    complex temp;
    temp.real=c1.real+c2.real;
    emp.imag=c1.imag+c2.imag;
    return(temp);
  }

  int main() {complex a(3.1, 4.5), b(2.3, 6.7);  // инициализация
    complex c;
    c=a+b;
    c.out();
    cin.get();
    return 0;
}
```

Результат выполнения



Для каждой комбинации типов операндов в переопределяемой операции необходимо ввести отдельную функцию, т.е. компилятор не может производить перестановку операндов местами, даже если базовая операция допускает это. Например, если необходима операция сложения комплексного и вещественного чисел:

```
complex a, c, d;
double b;
d = b + a;
```

то необходимо переопределить операцию сложения дважды:

```
friend complex operator + (complex, double);
friend complex operator + (double, complex);
```

1.5 Перегрузка операций индексирования и вызова функции

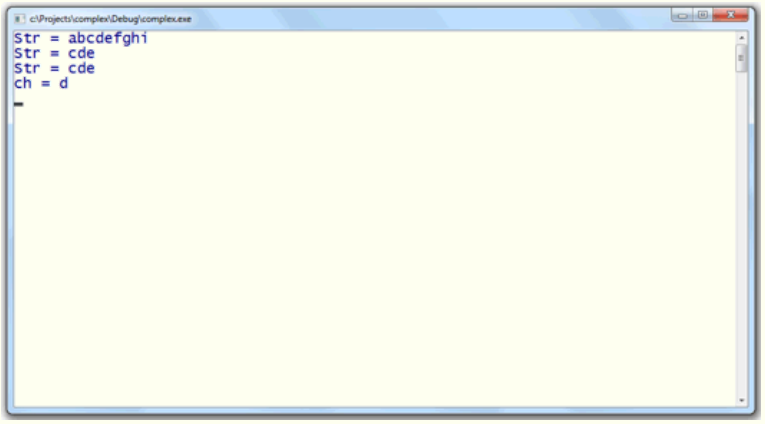
Переопределение операции **()** позволяет использовать синтаксис вызова функции применительно к объекту класса (имя объекта с круглыми скобками). Количество операндов в скобках может быть любым. Переопределение операции **[ ]** позволяет использовать синтаксис доступа к элементам массива (имя объекта с квадратными скобками).

```
  #include <string.h>
  #include <iostream.h>
  using namespace std;
  class String {                              // Строка переменной длины
    char *str;                                // Динамический массив символов
    int size;                                 // Длина строки
    public: String& operator()(int, int);  // Операция выделения подстроки
    char operator[](int);                     // Операция выделения символа
    void print(){
      if(str) cout << "Str = " << str << endl;
    }
    String (char* s = "") {
      size = strlen(s);
        str = new char [size+1];
        strcpy(str, s);
    }
    String (String& r) {
      str = new char [r.size];
      strcpy(str, r.str);
      size=r.size;
    }
  };
//------ Операция выделения подстроки -------------------
  String &  String:: operator()( int n1,  int n2) {
    size = n2-n1+1;
    char *tmp = new char [size+1];
    strncpy(tmp, (str+n1), size);
    tmp[size] = '\0';
    size++;
    delete [] str;
    str = new char [size];
    strcpy(str, tmp);
    delete [] tmp;
    return (*this);
  }

//------ Операция выделения символа -------------------
  char String::operator[](int index)  {
```

```
    return (str[index]);
  }
  int main() {
    String s1("abcdefghi");
    s1.print();
    String s2=s1(2,4);
    s2.print();
    s1.print();
    char ch = s2[1];
    cout << "ch = " << ch << endl;
    cin.get();
    return 0;
  }
```

Результат выполнения:



1.6 Перегрузка операции присваивания

Любой конструктор вызывается явно либо неявно в том случае, если необходимо создать новый объект какого-либо класса.

Рассматривая на примере создание нового объекта и его инициализацию, мы использовали конструктор копии. До проведения инициализации существовал только один объект. Второй был создан и инициализирован в результате работы конструктора копии. Однако, если бы существовало несколько объектов одного типа, и была бы необходимость в присвоении значений одного объекта элементам другого, то никакой из конструкторов не вызывается, так как объект уже был создан. При выполнении операции присваивания по умолчанию копирование значений происходит «поверхностно», но такое копирование не всегда допустимо. Например, недопустимо копирование массивов или указателей.

Если необходимо осуществить присваивание, но поведение операции присваивания по умолчанию не устраивает, то операция присваивания может быть перегружена.

Дополним приведенный выше пример реализации класса String перегруженным оператором присваивания:

```
String& operator=(const String& s) {
  if(size!=s.size) {
    size = s.size;
    delete[] str;
    str = new char [size];
   }
  strcpy(str, s.str);
  return (*this);
}

int main() {
  String s1("abcdefghi");
  s1.print();
  String s3;
  s3 = s1;
  s3.print();
  String s2=s1(2,4);
  s1.print();
  s2.print();
  s3.print();
  char ch = s2[1];
  cout << "ch =" << ch << endl;
  cin.get();
  return 0;
}
```

1.7 Перегрузка операций выделения памяти

Операции создания и уничтожения объектов в динамической памяти могут быть переопределены следующим образом

```
void *operator new(size_t size);
void  operator delete (void *);
```

где:

void * – указатель на область памяти, выделяемую под объект,

size – размер объекта в байтах,

size_t – тип размерности области памяти, int или long int.

Переопределение этих операций позволяет написать собственное распределение памяти для объектов класса.

Индивидуальные задания

2.1 Задание 1

№ варианта	Задание
1	Создать унарную перегруженную операцию ~, которая будет вычислять среднесуточную температуру.
2	Создать унарную перегруженную операцию ~, которая будет вычислять средний балл студента в сессию.
3	Создать унарную перегруженную операцию ~, которая будет вычислять среднюю скорость ветра за сутки.
4	Создать унарную перегруженную операцию ~, которая будет вычислять среднее количество пассажиров в сутки.
5	Создать унарную перегруженную операцию ~, которая будет вычислять среднее время выполнения рейса по машруту.
6	Создать унарную перегруженную операцию ~, которая будет вычислять среднее значение курса доллара ЦБ РФ.
7	Создать унарную перегруженную операцию ~, которая будет вычислять сумму выручки от продажи билетов.
8	Создать унарную перегруженную операцию ~, которая будет вычислять сумму выручки от продажи билетов.
9	Создать унарную перегруженную операцию ~, которая будет выдавать % набранных кандидатом голосов.
10	Создать унарную перегруженную операцию ~, которая будет выдавать TRUE если студент сдал все зачеты.
11	Создать унарную перегруженную операцию ~, которая будет вычислять среднесуточную температуру.
12	Создать унарную перегруженную операцию ~, которая будет вычислять количество набранных командами очков.

[В начало][Задание 1][Задание 2][Задание 3]

2.1 Задание 2

№ варианта	Задание
1	Создать бинарную перегруженную операцию %, которая будет вычислять отклонение среднесуточной температуры за две даты.
2	Создать бинарную перегруженную операцию %, которая будет вычислять отклонение среднего балла двух студентов.
3	Создать бинарную перегруженную операцию %, которая будет вычислять отклонение скорости ветра двух суток.
4	Создать бинарную перегруженную операцию %, которая будет вычислять разницу количества проданных билетов за двое суток.
5	Создать бинарную перегруженную операцию %, которая будет вычислять соотношение среднего времени выполнения рейса по машруту за две даты.
6	Создать бинарную перегруженную операцию %, которая будет вычислять отклонение в процентах курсов продажи и покупки доллара США банка от курса ЦБ РФ.
7	Создать бинарную перегруженную операцию %, которая будет вычислять процент от максимально-возможной выручки.
8	Создать бинарную перегруженную операцию %, которая будет вычислять процент от максимально-возможной выручки.
9	Создать бинарную перегруженную операцию %, которая будет выдавать TRUE если кандидат набрал более 51 % голосов.
10	Создать бинарную перегруженную операцию %, которая будет формировать массив дисциплин, по которым студент не сдал зачеты.
11	Создать бинарную перегруженную операцию %, которая будет вычислять отклонение среднесуточной температуры.
12	Создать бинарную перегруженную операцию %, которая будет вычислять команду-победителя.

[В начало][Задание 1][Задание 2][Задание 3]

2.1 Задание 3

№ варианта	Задание

1	Создать программу, которая будет выводить на экран отклонение среднесуточной температуры по двум одинаковым датам разных лет.
2	Создать программу, которая будет выводить на экран отклонение среднего балла по дисциплинам за два разных года.
3	Создать программу, которая будет выводить на экран отклонение скорости ветра текущего и прошлого годов в одну и ту же дату.
4	Создать программу, которая будет выводить на экран разницу количества проданных билетов в текущем и прошлом месяце.
5	Создать программу, которая будет выводить на экран соотношение среднего времени выполнения рейса по машруту в одни и теже дни разных недель.
6	Создать программу, которая будет выводить на экран курсы продажи и покупки доллара США ЦБ РФ и отклонение в процентах курсов продажи и покупки другого банка.
7	Создать программу, которая будет выводить на экран процент от максимально-возможной выручки за неделю.
8	Создать программу, которая будет выводить на экран процент от максимально-возможной выручки за неделю.
9	Создать программу, которая будет выводить список кандидатов в порядке убывания процента набранных голосов.
10	Создать программу, которая будет выводить на экран ведомости по дисциплинам со дисциплин, по которым студент не сдал зачеты.
11	Создать программу, которая будет выводить на экран отклонение среднесуточной температуры за один и тот же месяц прошлого и текущего годов.
12	Создать программу, которая будет выводить на экран перечень команд в порядке занятого ими места.

[В начало][Задание 1][Задание 2][Задание 3]