Московский авиационный институт

(технический исследовательский университет)

Кафедра 403

“Электронно-вычислительные средства и информатика”

Перегрузка операций

студент 1**-го курса Сергей Салов**

руководитель: **Мальшаков Г.В.**

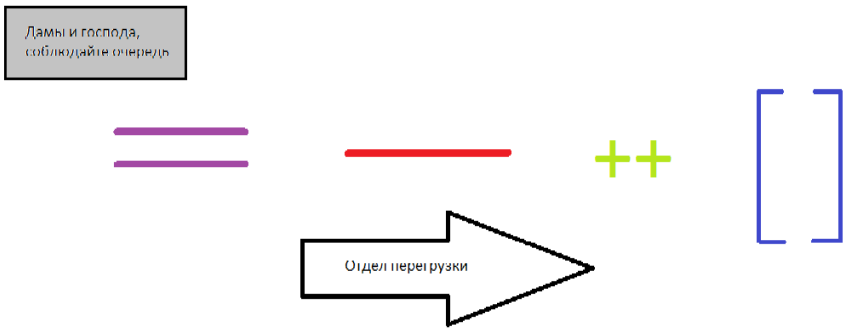
Москва, 2013

Для перегрузки операции в программе с использованием ключевого слова operator необходимо определить операторную функцию (operator function), задающую действия перегружаемой операции.

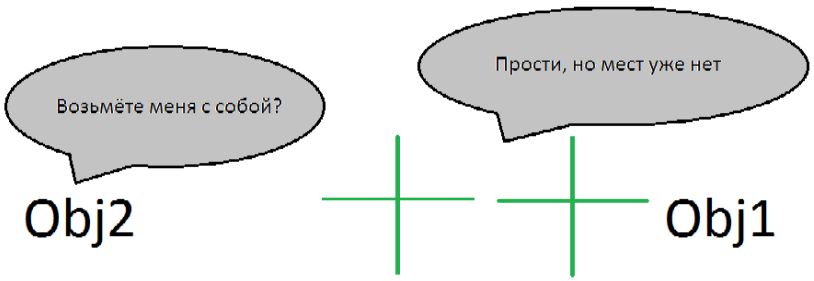
Формат операторной функции:

```
тип_возвр_значения  operator  знак_операции  ( список аргументов )
{
        // тело функции
}
```

При перегрузке операций:

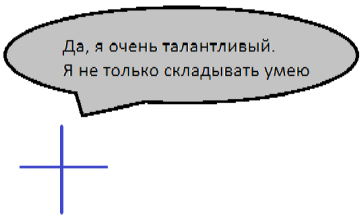


- невозможно изменить приоритет перегружаемых операций, который определен в таблице приоритетов языка С.



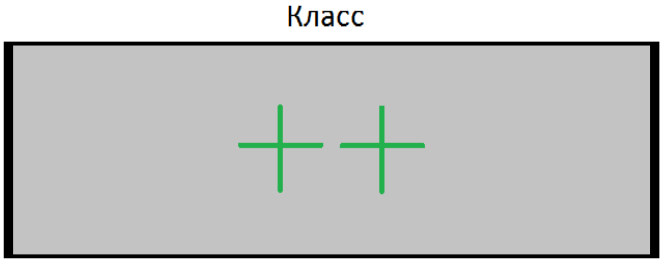
- невозможно изменить число операндов операций (определённых языком Си). Например, нельзя перегрузить оператор / так, чтобы в нем использовался только один операнд.

Переопределить возможно любые ***унарные*** и ***бинарные*** операции языка С++, за исключением: . -> :: ?: # sizeof

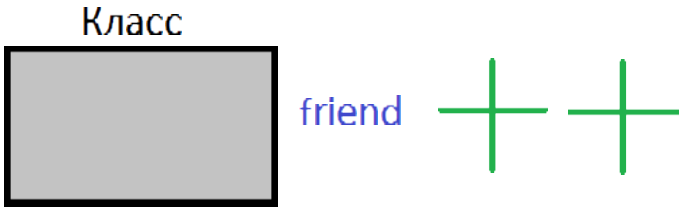


Выполняемые действия операцией определяются программистом внутри тела операторной функции, например, operator+(), и они могут быть любыми, к примеру не связанными со сложением.

Операторная функция может быть определена двумя способами:



- либо как компонентная функция без параметров (определяется внутри класса, как его метод),



- либо как простая (глобальная, возможно дружественная классу) функция с одним параметром (определяется вне класса как обычная функция).

Перегрузка унарных операций (содержащих один операнд)

Унарная операция ⊕ может быть определена двумя способами: либо как компонентный метод без параметров, либо как простая (глобальная, возможно дружественная классу) функция с одним параметрами.

Тип реализации	Синтаксис описания функции	Примечание
компонентная (как метода)	описывается в области класса: тип_возвр_значения operator ⊕ ()	в качестве операнда выступает вызываемый объект класса
простая (как функции)	описывается вне области класса: тип_возвр_значения operator ⊕ (тип_объекта)	операнд в функцию передается через параметр

⊕ - знак операции.

Примеры

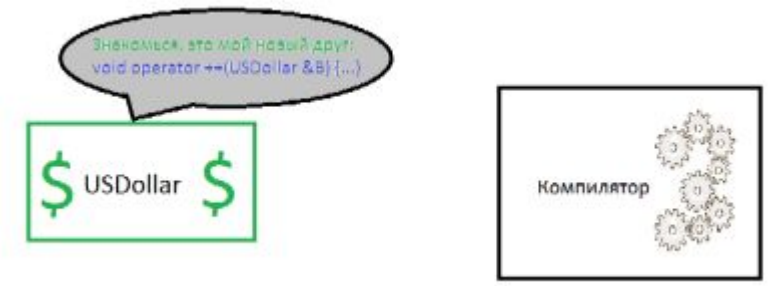
1) Перегрузка унарной операции ++ в виде компонентной операторной функции

```
class USDollar  {
          unsigned int Dollars;
          unsigned int Cents;
          ...
     public:
          ...
          void operator++() {
                ++Cents;
                if (Cents=100){++Dollars; Cents=0;}
          }
};

void main() {
       USDollar Bank;
       ++Bank;                          // вызов перегруженной операции
}
```

2) Перегрузка унарной операции ++ в виде простой операторной функции

```
class USDollar {
          unsigned int Dollars;
          unsigned int Cents;
          ...
    public:
          ...
          friend void operator++(USDollar&);
};
```

```
void operator++(USDollar& ob) {
        ++ob.Cents;
        if (ob.Cents=100){++ob.Dollars; ob.Cents=0;}
}

void main() {
       class USDollar Bank;
       ++Bank;                              // вызов перегруженной операции
}
```



При перегрузке операции в виде простой операторной функции требуется объявлять функцию void operator ++(USDollar &B) {…} дружественной классу USDollar, используя ключевое слово friend, чтобы ей стали доступны закрытые поля Dollars и Cents.

Инкремент и декремент могут быть как префиксными, так и постфиксными. Для «превращения» перегруженной операции в постфиксную необходимо добавить в список параметров ключевое слово int, говорящий об её постфиксности компилятору.

Примеры:

1) Постфиксный декремент -- в виде компонентной операторной функции

```
class USDollar  {
            ...
      public:
            ...
            USDollar operator--(int) {
                        USDollar A;
                        A.Dollars=Dollars; A.Cents=Cents
                        if (Cents=0){--Dollars; Cents=100;}
                        --Cents;
                        return A;
                       }
};

void main() {
        USDollar Bank;
        Bank--;                             // вызов перегруженной операции
}
```

2) Постфиксный декремент -- в виде простой операторной функции

```
class USDollar {
        ...
    public:
        ...
        friend USDollar operator--(USDollar&, int);
};
USDollar operator--(USDollar& ob, int) {
          USDollar A;
          A.Dollars=ob.Dollars; A.Cents=ob.Cents
          if (Cents=0){--Dollars; Cents=100;}
          --Cents;
          return A;
}
```

```
void main() {
      class USDollar Bank;
      Bank--;                          // вызов перегруженной операции
}
```

Задания на проверку

Дан класс Atom

```
class Atom {
   int P;       // число протонов
   int E;       // число электронов
   int N;       // число нейтронов
};
```

Перегрузите для данного класса следующие операции:

1. Постфиксный инкремент ++ в виде простой операторной функции, увеличивающий число электронов объекта на единицу:

2. Префиксный декремент -- в виде компонентной операторной функции, уменьшающий число протонов на 4 и число нейтронов на 2:

Перегрузка бинарных операций (содержащих два операнда)

Бинарная операция ⊕ может быть определена двумя способами: либо как компонентный метод с одним параметром, либо как простая (глобальная, возможно дружественная классу) функция с двумя параметрами.

Тип реализации	Синтаксис описания операторной функции	Примечание
компонентная (как метода)	описывается в области класса: тип_возвр_значения operator ⊕ (тип_объекта)	В качестве первого операнда выступает вызываемый объект класса, в качестве второго - входной параметр.
простая (как функции)	описывается вне области класса: тип_возвр_знач operator ⊕ (тип_об1, тип_об2)	Операнды в функцию передается через параметры. Один из параметров обязательно должен иметь не стандартный тип данных, иначе перегрузка операции невозможна.

⊕ - знак операции.

Примеры

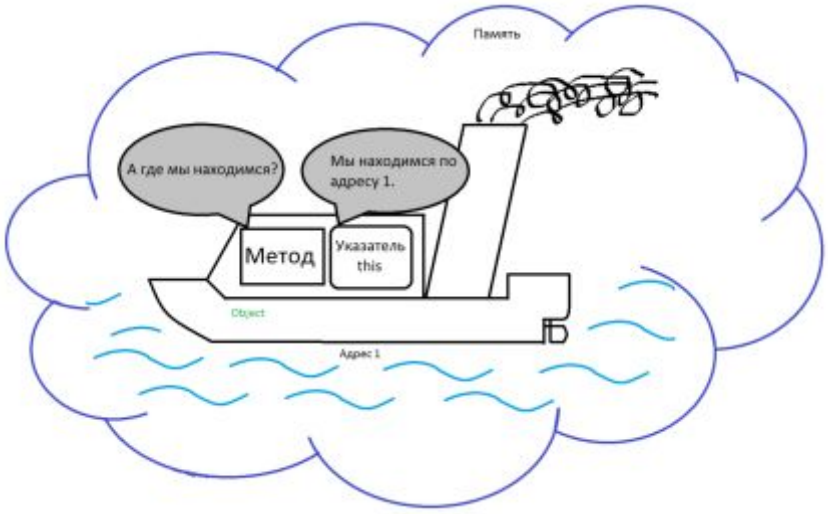
1) Перегрузка бинарной операции + в виде компонентной операторной функции.

```
class USDollar  {
        unsigned int Dollars;
        unsigned int Cents;
    public:
        USDollar (unsigned int Dollars, unsigned int Cents) {
                this->Dollars=Dollars; this->Cents=Cents;
```

```
        }
         USDollar operator+(USDollar &B) {
                Dollars+=B.Dollars; Cents+=B.Cents;
                if (Cents>=100) {++Dollars; Cents-=100;}
                return *this;
         }
};
void main() {
   USDollar Bank (0, 0), BusinessMan (200, 53), BusinessWoman (500, 47);
   Bank=BusinessMan+BusinessWoman;
   return 0;
}
```

2) Перегрузка бинарной операции + в виде простой операторной функции

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

USDollar operator+(USDollar& B ,USDollar& C) {
            USDollar A;
            A.Dollars=B.Dollars+C.Dollars; A.Cents=B.Cents+C.Cents;
            if (A.Cents>=100){++A.Dollars; A.Cents-=100;}
            return A;
}

void main() {
   USDollar Bank (0, 0), BusinessMan (200, 53), BusinessWoman (500, 47);
   Bank=BusinessMan+BusinessWoman;
   return 0;
}
```



Внутри операторной функции перегрузки операции «+» для класса USDollar используется неявный константный указатель this, равный адресу объекта текущего класса. Его можно использовать для возврата содержимого объекта, на который указывает this (как приведено в примере), а также для обращения к полю, перекрытому параметром (если входной параметр имеет такое же имя, как и поле объекта класса):

```
class A {
  int a;
  A (int a) {this->a=a;}
};
```

Перегрузка операции индексации []:

1) В виде компонентной операторной функции

```
class person{...};
class adresbook {
    ...
    public:
      person&  operator[](int);                    // доступ к i-му объекту
};

person& adresbook :: operator[](int i){...}

void main() {
   adresbook persons;
   person record;
       ...
   record = persons [3];
}
```

2) В виде простой операторной функции

```
class person{...};
class adresbook {
     ...
    public:
      friend person& operator[](const adresbook&,int);       // доступ к i-му объекту
};

person&  operator[](const adresbook& ob ,int i) {...}

void main() {
   adresbook persons;
   person record;
       ...
   record = persons [3];
}
```

Перегрузка операции вызова функции (). Класс с такой перегрузкой называется функциональным.

Пример:

```
class if_great {
  int operator () (USDollar A, USDollar B) { return (A.Dollars>B.Dollars);}
};
```

```
void main {
  USDollar Bank1, Bank2;
  Bank1.Dollars=100;
  Bank2.Dollars=99;
  If_great B;
  Cout<<C(Bank1,Bank2); // на экран будет выведено 0;
```

Перегрузка операций ввода-вывода (>> ,<<):

Пример:

```
class USDollar  {
        unsigned int Dollars;
        unsigned int Cents;
};
void USDollar::operator << (const USDollar &B) {
      cout<<B.Dollars”.”B.Censt/100;
}
```

Перегрузка логической операции <=

Пример:

```
class USDollar  {
        unsigned int Dollars;
        unsigned int Cents;
      public:
        USDollar (unsigned int Dollars, unsigned int Cents) {
                this->Dollars=Dollars; this->Cents=Cents;
        }
        int operator <= (const USDollar &B) {
            if (Dollars<=B.Dollars) {return 1;}
            else {return 0;}
        }
};

Void main () {
      USDollar Bank1 (1000, 35), Bank2 (4500, 77);
      int A;
      A=(Bank1<=Bank2);
      return 0;
}
```

Преимущества и недостатки простой и компонентной операторных функций

	Достоинства	Недостатки
Компонентная	+ Операции [], (), = могут быть перегружены + Удобный перенос функций	- Невозможно перегрузить операцию, первым оператором которой является стандартный тип (например 5**объект производного типа*) - Невозможно перегрузить операции для объектов следующих типов: o enum o struct o union
Простая	+ Возможно перегрузить операцию, первым оператором которой является стандартный тип (например 5**объект производного типа*) + Возможно перегрузить операции для объектов следующих типов: o enum o struct o union	- Операции [], (), = не могут быть перегружены - Неудобный перенос функций

Задания на проверку

Дан класс Library

```
class Library {
    unsigned char Number; // номер библиотеки
    unsigned  int Books; // число книг
    unsigned short Readers; // число читателей
};
```

Перегрузите для данного класса следующие операции:

1. Вычитание – в виде простой операторной функции, результатом которой является новый объект с именем первой библиотеки, а также с разницей числа книг и читателей исходных библиотек.

2. Сложение + в виде компонентной операторной функции, результатом которой является первая библиотека, имя которой было изменено на имя второй, число книг и читателей сложено с числом книг и читателей второй.

3. Сравнение >= в виде компонентной операторной функции, результатом которой будет истина, если число читателей первой библиотеки больше или равно числу читателей второй, и ложь в обратном случае

Контрольные задания

1. Найдите ошибки

```
class El {
    unsigned int Num;              // номер элемента
    int Temp;                      // температура плавления
    int Ten;                       // плотность
    El (unsigned int x, int y, int z) {Num=x; Temp=y; Ten=z;}
  public:
    friend void operator --(El &);
    friend El operator = (El &, El &);
    void operator+(int x) {Num+=x;}
    int operator ->(El& B) {B.Temp+=Temp;
                                    return B;}
    void operator <() {if (B.Ten<Ten)
                            return B;
                        else
                            return this;}
    El operator ++() {++Temp;
                        ++Ten;}
};

El operator = (El &A, El &B) {
        A.Temp=B.Temp;
}

El operator *(El &A, El &B) {
        B.Ten*=A.Ten;
        return B;
}

Void main() {
  El X(1, 1000, 53), Y(23, 3700, 78), Z(0, 0, 0), A(72, 353, 121), B(8, 999, 290), C(100, 2200, 111);
    X=X+5;
    Z=X->Y;
    X=Y*Z;
    B=A<X++;
    --C;
    Y+5;
    Return 0;

}
```

2. Разработайте класс книга, имеющий название, число страниц, год издания и стоимость. Перегрузите для этого класса следующие операции:

- префиксный инкремент ++ в виде компонентной операторной функции, увеличивающий стоимость книги на 10 рублей.
- Постфиксный декремент – в виде простой операторной функции, уменьшающий стоимость книги на 15 рублей.
- Логическое сложение | в виде компонентной операторной функции, результатом которой является первая книга с именем второй, с арифметически сложенным числом страниц, со своим годом издания и с логически сложенной стоимостью. (Вывод объекта осуществлять, используя указатель this)
- Умножение * в виде простой операторной функции, результатом которой является новый объект с именем и годом издания второй книги, с умноженным числом страниц и стоимости.
- Сравнение < в виде простой операторной функции, результатом которой будет являться первая книга, если её стоимость меньше второй, и вторая книга в обратном случае.
- Присваивание = в виде компонентной операторной функции, результатом которой будет присваивание первой книге имени второй.

Словарь терминов

Переменная – это именованная область памяти, предназначенная для хранения данных определённого типа.

Операция — это конструкция в языках программирования, аналогичная по записи математическим операциям, то есть специальный способ записи некоторых действий.

Операнд – это аргумент операции, над которым выполняется операция

Унарная операция – это операция, совершаемая только над одним операндом.

Бинарная операция – это операция, совершаемая над двумя операндами.

Стандартные типы данных – это типы данных, определённые разработчиками языка программирования Си (к примеру short, long и т.д.).

Производные типы данных – это типы данных, определённые программистом.

Перегрузка операции – это переопределение операций среды программирования (+, -, *, / и т.д.) для производных типов данных.

Функция — это фрагмент кода или алгоритм, реализованный на каком-то языке программирования, с целью выполнения определённой последовательности операций.

Операторная функция – это функция, при объявлении которой используется специальное имя, состоящее из ключевого слова operator с символом операции, предназначенная для переопределения действий указанной операции.

Класс - это тип данных, содержащий поля и методы для описания существующих объектов.

Объект – это переменная с типом класса, имеющая свои свойства, поля и методы.

Метод – это процедура внутри класса.

Компонентная операторная функция – это функция, объявленная внутри класса.

Простая операторная функция – это функция, объявленная вне класса как общая функция.

Указатель – это переменная хранящая адрес ячейки памяти.

this – это неявный константный указатель, равный адресу объекта текущего класса.