acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . OOP-Instrumental_a :

Перегрузка методов. Перегрузка операторов.

Перегрузка методов (функций)

В C++ в одном пространстве имен можно переопределять функциональность методов с одним и тем же именем, но с разным набором параметов.

Рассморим задачу

Дробь задается двумя положительными целыми числами, не превышающими $2^{32} - 1$. Требуется реализовать инструментарий работы с дробями.

Сначала реализуем возможность задавать дробь сразу при ее создании и различные методы печати этой дроби.

```
#include <iostream>
#include <cstdlib>
using namespace std;

class Drob{
 int cel, chisl,znam;

public:
  Drob(); // конструктор дроби по-умолчанию
// Инициализирующий конструктор
  Drob(int,int,int);
// Устанавливаем значение
  void setVol(int, int,int);
// Печать дроби в формате простой дроби
   void print();
// Печать дроби в десятичном виде (n - количество знаков после запятой)
   void print( int n);
};
// Конструктор "по-умолчанию"
Drob::Drob(){
  cel = chisl = 0;
  znam = 1;
};
// Инициализирующий конструктор (с параметрами )
// По количеству и типам параметров C++ различает какую из
// какую из имеющихся функций нужно применять
Drob::Drob(int c, int ch, int z){
     cel = c;
     chisl = ch;
     znam = z;
};

// "Обычный" print()
void Drob::print(){
   cout<<cel<<'('<<chisl<<'/'<<znam<<')';
};

// print() "с парамером" - n означает сколько знаков
// после запятой нужно напечатать.
void Drob::print( int n){
   int c = cel * znam + chisl;
   int b = znam;
  for(int i = 0; i < n + 1; i++){
   int pr = c / b;
   cout<<pr;
   if (i == 0)
```

```
    cout<<'.';
   int next = c % b;
   next *= 10;
   while(next < b){
     next *= 10;
     i++;
     if (next >= b || i>=n) break;
     cout<<'0';
   }
  c = next;
 }
  cout<<endl;
};

// Примеры использования конструктора и переопределенных
// операторов
int main(){
  Drob a; // конструктор "по-умолчанию"
  Drob b(1,2,3); // инициализирующий конструктор
  a.print();// "обычный" print()
  b.print(7);// print() "с парамером" (7 знаков)

 return 0;
}
```

При работе с математическими объектами и в ряде других случаев перегрузка операторов очень полезна, так как для зарезервированный в языке операторов сохраняются все их свойства: разбор скобочных структур, приоритет и т.д.

Задачи

Задача 1

Реализовать для работы с простыми дробями необходимо:

1. исправить печать дроби в формате ***<целая часть>(<числитель><знаменатель>)***. Если цела часть отсутствует, она не печатается, если знаменатель равен **1**, печатается как целое, если отсутствует дробная часть или числитель равен **0**, дробная часть не печатается. Дробь должна быть представлена в несократимом виде.
2. реализовать ***void Drob::print(char);*** - если указана ***** - печать числа в стандартном виде: целая часть предсавляется одной цифрой, далее точка, далее непереодическая часть дроби, затем периодическая в скобках, знак умнжения **('x')** и десятичная степень, на которую нужно умнжить число, чтобы получить правильное. Например, представление дроби ***5/7***: ***0.(714285)x0***, то есть $0.(714285)x10^0$

Переопределение операторов

Продолжим рассмаривать задачу с дробями.

В C++ можно переопределять сществующие в языке операторы: **+, - , /, *, ==, !** и др. Операторы бывают: "бинарные"; "унарные" - "префиксные", "постфиксные" и др.

Дроби, очевидно, удобнее складывать как обычные числа. постараемя переопределить операторы **+, ++** - увеличение на **1** и **!** - получение обратоной дроби.

```
#include <iostream>
#include <cstdlib>
using namespace std;

class Drob{
 int cel, chisl,znam;

public:
  Drob(); // конструктор дроби по-умолчанию
// Инициализирующий конструктор
  Drob(int,int,int);
// Устанавливаем значение
  void setVol(int, int,int);
/*
Зарезервированное слово "operator" обозначает описание именно
оператора, а не какой-либо другой функции.
В C++ существуют два способа передачи адреса: указатель (*) и
ссылка (&).
  При передачи указателя синтаксис работы с этим объектом
как с указателем (разыменовывание - *point, обращение к
атрибутам и методам - point->metod()).

  При передачи ссылки сохраняется синтаксис как при работе с
```

```
самим объектом: разыменовывание не нужно.

Но объявить ссылку как отдельный объект нельзя. Она
используется только для передачи данных в функции и как
возвращаемые значения функций.

При передачи адреса есть опасность несанкционированно
изменить значение объекта. Чтобы этого не произошло,
объявляется const. Тогда такие попытки будут отловлены на
стадии компиляции.

*/
  Drob operator+(const Drob&);

/*
  Оператор ++ - постфиксный, то есть пишется после объекта.
Поэтому для соблюдений правил синтаксиса ему нужно указать
неиспользуемый параметр int

Так как этот оператор изменяет текущий объект, он возвращает
указатель на себя самого. Для того, чтобы иметь указатель на текущий
объект, имеется специальное слово this.
*/
  Drob& operator++(int);


/*
  Оператор ! - префиксный, то есть пишется перед объектом.
Ему не нужно указывать дополнительный параметр, так как сам
объект формально является его параметром.

  Для всех операторов существует строго определенное количество
параметров, а для некоторых параметров еще должен сохраняться его
тип. Переопределяя оператор, нельзя изменить количество параметров
для этого оператора (а для некоторых еще и тип). Переопределению
подлежит, в основном, функциональность.
*/

  Drob& operator!();

// Для отладки
  void print();
  void print(char);
  void print(int);
};

Drob::Drob(){
  cel = chisl = 0;
  znam = 1;
};

void Drob::print(){
   cout<<cel<<'('<<chisl<<'/'<<znam<<')';
};

void Drob::setVol(int a, int b, int c){
     cel = a;
     chisl = b;
     znam = c;
};

/*
 Это "заглушка" - то есть функциональность
реализована не полностью.
Этот метод  - только как пример написания подобных.

При использовании параметра, переданного "по-ссылке" синтаксис
использования такой же как и для параметров, переданных как копии
*/
Drob Drob::operator+(const Drob& a){
Drob tmp;
     tmp.chisl = chisl + a.chisl;
     tmp.znam = znam;
     return tmp;
};

/*
  Это также "заглушка".
Изменения произошли с текщим объектом, и
```

```
возвращается указатель на текущий объект.
  Указывается параметр int, который не используется.
*/

Drob& Drob::operator++(int){
   cel++;

   return *this;
};

/*
  Префиксный оператор.
Также возвращается ссылка на текущий объект.
*/
Drob& Drob::operator!(){
    int z = chisl + cel;
    chisl = znam;
    znam = z;
    cel=0;
    return *this;
};

int main(){
  Drob c;
  Drob a(0,2,5), b(0,1,5);


// Пример использования оператора +
  c= a + b;
// Выше была показана краткая запись (рекомендуется)
// Полный вызов выглядит так: c = a.operator+(b);

  c.print();
// Пример использования инкремента.
  c++;
  c.print();
// Пример использования инвертирования.
  !c;
  c.print(*);

}
```

Задача 2 Восхождение

Крокодил Гена и Чебурашка совершают восхождение на гору. Чебурашка едет в рюкзаке за спиной у Гены. Гена все время падает и скатывается вниз из-за Чебурашки. Он рассказывает смешные анекдоты. Перед началом восхождения Гена тоже упал. После того, как они проходят 1/n часть расстояния до вершины, Чебурашка рассказывает Гене анекдот, и тот от смеха вместе с Чебурашкой скатывается вниз на 1/k часть пройденного расстояния от того места куда он в прошлый раз скатился. И всякий раз, как только Гена поднимается на 1/n часть расстояния отделяющего место, куда он скатился, от вершины, Чебурашка рассказывает очередной анекдот, и Гена вновь скатывается на 1/k пройденного с этого места пути. Последний (m-ный) анекдот Чебурашка рассказал, когда до вершины остался 1 метр, но этот анекдот оказался настолько смешным, что на этот раз Гена скатился к самому подножью горы, и на этом восхождение закончилось. Написать программу, которая вычисляет высоту горы и выдает ТОЧНЫЙ результат в виде смешанной дроби, не округляя и не приводя ее к десятичной. Дробная часть записывается в круглых скобках, числитель отделяется от знаменателя косой чертой. Например 102(71/93) обозначает высоту горы

Задача 3 Калькулятор

Дан заголовочный файл, описывающий представление числа в калькуляторе. Реализовать и проверить работу всех его функций

```
#include <iostream>
#include <cstdlib>
/*
 В калькуляторе ЭлектроникаВ3-21 реализованны только операции с плавающей точкой
 При этом само число (как результат ввода, так и результат операций) отображается
  следующим  образом.
 1. Всего значащих цифр в числе не более 7 и не менее 1
 2. Если сумма цифр в целой и дробной части не превышает 7, а порядок числа не более 7,
    то отображается цела часть, точка, затем дробная часть. Десятичная степень числа (порядок)
    указывается как 0 (формат %02d)
 3. Если порядок числа меньше 1 и больше 6, то в целой части отображается одна цифра,
```

```
    затем точка, затем дробная часть. Порядок числа (pow) вычисляется таким образом,
    чтобы при умножении отображаемого числа на 10^pow получалось бы исходное число .
 4. Если абсолютное значение дробной части числа <0.0000001, число отображается как целое
 5. Если число < 0, перед ним ставится знак -
 6. Если в вычислениях получено число, у которого больше 7 значащих цифр, последние
    цифры округляются до 7 и, соответственно, вычисляется порядок

*/
/*
  При реализации учесть что некоторые операции не могут выполниться. Использовать конструкцию
   try...catch для разрешения ситуации
*/
class CalcNumber{
// Здесь явно переизбытьчная информация и числе.
// Это сделано для облегчения реализации кода
// Если какие-то атрибуты не будут использованы, не страшно
// За отображение и выполненеие действий этот класс отвечает полностью
   int integer; // целая часть числа
   unsigned int fract;  // дробная часть числа
   char pow;  // порядок числа
   float base; // собственно число для вычислений
/*
  Число перелставляпется строкой текста в формате:
   <знак><целая часть>.<дробная часть><знак порядка><порядок>
  На все число - 13 символов.
   Знак числа указывается, если число < 0.
  Дробная часть может присутствовать или отсутствовать
   Порядок может указываться или отсутствовать
  Знак порядка указывается только если порядок меньше 0
  Например,
   1234
  -1234
  123.4567
   1234       10
   1.234567  -99
   -1.234567  99
*/
   String text_moda;
// состояние числа
/*
   SW  = 1 ввод числа (целая часть) (цифры типа 10 * n + 4, n - цифра)
   COMMA = 3 ввод дробной части числа после получения кода 46
   POW = 5 ввод степени числа (степень не менее -99 и не более 99) и
               смена знака степени
*/
   unsigned char moda;
   public:
// число представлено как 0
      CalcNumber();
/*
  Число вводится строкой текста в формате:
  <знак><целая часть>.<дробная часть><знак порядка><порядок>
  На все число - 13 символов.
   Знак числа указывается, если число < 0.
  Дробная часть может присутствовать или отсутствовать
   Порядок может указываться или отсутствовать
  Знак порядка указывается только если порядок меньше 0
  Например,
   1234
  -1234
  123.4567
   1234       10
   1.234567  -99
   -1.234567  99
*/
      CalcNumber(char*);
// Получение нового кода при вводе числа
/*
   Функция получает код:
   Если код в формате 10 * n + 4, то это код цифры n
   Если код 46 - введена запятая, число не изменяется, но
   следующие цифры входят в состав дробной части
   Если изменяется целая или дробная часть, то получение
        кода 56 - смена знака,
   меняется знак у числа
   Если код 66 - ввод степени (характеристики),
        цифры составляют число - степень, ввод кода знака
   в этом случае меняет знак степени
   Если общее количество цифр для значащей части числа > 7 или
```

```
       для степени > 2, ввод игнорируется.
*/
      CalcNumber& addDig(unsigned char kod);
// перегрузка операторов
// Результат работы ВСЕХ опрераторов изменяет текущее число

// Бинарные операторы
      CalcNumber& operator+=(const CalcNumber&); // сумма чисел
      CalcNumber& operator-=(const CalcNumber&); // разность чисел
      CalcNumber& operator*=(const CalcNumber&); // произведение
// frst - ПЕРВЫЙ аргумент деления, результат деления изменяет текущее число
      CalcNumber& operator/=(const CalcNumber& frst); // деление
// текущее число возводится в степень аргумента
      CalcNumber& operator^(const CalcNumber&);
// Унарные операторы
      CalcNumber& exp(const CalcNumber&);// exp
      CalcNumber& sin();
      CalcNumber& cos();
      CalcNumber& sign(); // изменение знака числа
      CalcNumber& square(); // квадратный корень из числа
      CalcNumber& sqrt(); // квадрат числа
      CalcNumber& pi(); // число пи
      CalcNumber& ln(); // натуральный логарифм
// операции сравнения возвращают 1, если выполняется и 0, если нет
      bool operator>(const CalcNumber&);
      bool operator<(const CalcNumber&);
      bool operator>=(const CalcNumber&);
      bool operator<=(const CalcNumber&);
// операции присваивания
    CalcNumber operator=(const CalcNumber&);
// печать числа:
// Формат печати см. выше
      void show();
};
```

-- TatyanaOvsyannikova2011 - 17 Feb 2016