acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . OOP-Herstring :

В качестве примера реализации наследования от библиотечного класса рассмотрим класс **Word** - наследник **string.**

Наследование позволит в полной мере использовать все функции класса ***string*** и позволит решать дополнительные задачи, где требуется дополнительная функциональность

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <stdlib.h>
#include <set>

using namespace std;

// Расширенные возможности работы со строками
// Наследует стандартному классу string
// ВСЕ доступные функции класса string становятся доступны и оъъектам класса Word
class Word:public string{

  int nbk; // количество различных букв в слове
// "внутренние" функции класса Word
// подсчет различных букв в слове
  int count();
// сортировка содержимого строки (для внутреннего использования)
  void sort();

public:
//Конструктор по умолчанию
  Word();

// Инициализация строкой
  Word(string&);
// Инициализация  с-строкой
  Word(char*);
// инициализация символом
  Word(char);

// Операторы присваивания.
// Необходимо переопределять, так как C++ "не знает"
// что делать с дополнительным атрибутом nbk
  Word* operator=(string);
  Word* operator=(const char*);
  Word* operator=(char);
// Все операторы, уже реализованные в строках string работают в этом классе
//_______Дополнительные операторы:_____________________

// Оператор * - увеличивает строку в указанное число раз:
// s=abc; c=s*3 -> c=abcabcabc
  Word& operator*(int);

//Оператор инвертирования
// "переворачивае" слово: abcd -> dcba
 Word& operator!();
```

```
// Оператор /
// Ищет период с в слове, и , если слово состоит из подстрок,
// периодически встречающихся указанное число раз, уменьшает
// слово до размера этого периода
// если это не так, слово не изменяется
// Например:
// aaaaa/5 = a
// abababab/2=abab
// abababab/3=abababab
   Word& operator/(int);

// Оператор - : ищет ВСЕ, совпадающие буквы с строках,
// удалаяет их из "нашей" строки и строки параметра и возвращает
// строку, состоящую из этих букв (сортированную)
// 12135 - 351 ->12; пустая; результат 135
// или 212321 - 513 -> 2122; 5; 13
  Word& operator-(Word&);
// Заменяет найденные буквы (не важно в каком порядке были) на подстроку
// и сортирует позвращаемое слово
  void instead(Word&,Word&);

// возвращает количество раздичных букв в слове
  int getLet();
};

Word::Word(){
// сначала работает конструктор string по умолчанию
// затем наши действия
  nbk=0;

};
// Здесь нужно указать чем и как инициализировать конструктор базового класса
Word::Word(string& a):string(a){
  sort();
  nbk=count();

};
int Word::getLet(){
   return nbk;
};
int Word::count(){
   // реализовать функцию подсчета букв
};

Word* Word::operator=(string a){
// чтобы работать с атрибутом строки, который
// входит в наш класс, но сам по себе недоступен,
// преобразуетм объекты типа Word к типу string
// для string все операторы присваивания уже реализованы
    (*((string*)this))=string::operator=(a);
    sort();
    nbk=count();
    return this;
};

Word* Word::operator=(const char* a){
    (*((string*)this))=string::operator=(a);
    sort();
    nbk=count();
    return this;
};
Word* Word::operator=(char a){
    (*((string*)this))=string::operator=(a);
    nbk=1;
    return this;
};
int cmp(const void* a, const void* b){
   return *(char*)a - *(char*)b;
};
// пример сортировки
void Word::sort(){
```

```
   qsort((void*)(*(string*)this).data(),(*(string*)this).length(), sizeof(char),cmp);
};

// Реализовать;
Word& Word::operator-(Word& a){
};
Word& operator*(int){

};
Word& operator/(int){
};

int main(){
  string z="231116";
  Word a,b,c;
  a=z;
  b="613";
// проверить как работает оператор+
  c=a+b;
  cout<<c<<endl;

 return 0;
}
```

Задача 1

Проверить правильно ли работает оператор **+** в классе **Word.** Если неправильно, переопределить свой.

Задача 2

Реализовать ВСЕ функции, описанные в интрефейсе. Добится правильной функциональности операторов.

Задача 3 (1)

Даны два слова **top** и **AwA.** Написать программу, которая получает из них вот такое слово:

toppotAwAtoppotAwAtoppotAwAAwAtoppotAwAtoppotAwAtoppot

Использовать класс **Word.** Код программы (main) должен быть как можно короче.

Задача 3 (2)

Для корректной верстки все слова в тексте должны быть раздены ТОЛЬКО одним пробелом. Между словом и знаками препинания (**.**, **,**, **!**, **:**) пробелов быть не должно, но, за знаком должен быть ОДИН пробел.

Используя класс **Word** написать как можно более котрткую программу (main), которая исправляет текст, в котором пропущены или проставлены лишние пробелы. Если пробел пропущен между словами без знаков препинания, считать это одним словом.

Задача 4

В строке могут встречать одинаковые подстроки. Они могут быть расположены произвольно. Например: ***abc123wer123*** или ***abc123wer123abc***

Требуется написать программу, которая ищет самые длинные повторяющиеся подстроки, подсчитывает из количество и "сворачивает" строку такми образом:
abc123wer123 -> ***2(123)abcwer***
или
abc123wer123abc -> ***2(123)2(abc)wer*** -> 2(123abc)wer

То есть одинаковые "множители" "выносятся за скобки". Символы внутри подстроки сортируются, Найденные подстроки сортируются по "множителю". Остальная часть не подлежит преобразованию:
12w12abce1212abcr -> ***2(abc)4(12)wer***

В решении использовать класс **Word**

-- TatyanaOvsyannikova2011 - 28 Oct 2015