МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ М.В. ЛОМОНОСОВА

Факультет
вычислительной математики
и кибернетики

Ю.С. Корухова

СБОРНИК ЗАДАЧ И УПРАЖНЕНИЙ ПО ЯЗЫКУ C++

Москва

2009

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА
Факультет вычислительной математики и кибернетики

Ю.С. Корухова

СБОРНИК ЗАДАЧ И УПРАЖНЕНИЙ ПО ЯЗЫКУ С++

Учебное пособие
для студентов-бакалавров II курса,
обучающихся по направлению
«Информационные технологии»

МАКС ПРЕСС

МОСКВА – 2009

УДК 004.43(075.8)
ББК 32.973-018.1я73
К66

Печатается по решению Редакционно-издательского совета факультета вычислительной математики и кибернетики Московского государственного университета имени М.В. Ломоносова

Рецензенты:
доцент *Н.Н. Попова;* доцент *С.Б. Березин*

Корухова Ю.С.

К66 **Сборник задач и упражнений по языку C++:** Учебное пособие для студентов-бакалавров II курса, обучающихся по направлению «Информационные технологии». – М.: Издательский отдел факультета ВМиК МГУ имени М.В. Ломоносова (лицензия ИД N 05899 от 24.09.2001 г.); МАКС Пресс, 2009. – 24 с.
ISBN 978-5-89407-387-3
ISBN 978-5-317-03053-7

В сборнике представлены задачи, используемые при изучении языка C++ на занятиях практикума на отделении подготовки бакалавров по направлению «Информационные технологии» факультета ВМК МГУ. Рассматривается версия языка C++, соответствующая ANSI-стандарту.

В задачах рассматриваются принципы построения абстрактных типов данных, описание конструкторов, деструкторов классов, определение перегруженных операций и функций, создание иерархий классов и использование виртуальных функций, генерация и обработка исключений, шаблоны классов и функций. В последнем разделе представлены задачи, выполнение которых предполагается на ЭВМ в операционной системе семейства UNIX.

Сборник составлен с учетом опыта преподавания программирования на факультете вычислительной математики и кибернетики МГУ. Для студентов факультета ВМК, изучающих язык C++, а также для преподавателей и аспирантов.

УДК 004.43(075.8)
ББК 32.973-018.1я73

Автор выражает огромную благодарность Т.В. Руденко за помощь в подготовке сборника

Учебное издание

КОРУХОВА Юлия Станиславовна
СБОРНИК ЗАДАЧ И УПРАЖНЕНИЙ ПО ЯЗЫКУ C++

Учебное пособие

Издательский отдел Факультета вычислительной математики и кибернетики
МГУ им. М.В. Ломоносова
Лицензия ИД N 05899 от 24.09.01 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ имени М.В. Ломоносова, 2-й учебный корпус

Напечатано с готового оригинал-макета
Издательство ООО «МАКС Пресс»
Лицензия ИД N 00510 от 01.12.99 г. Подписано к печати 02.12.2009 г.
Формат 60x88 1/16. Усл.печ.л. 1,5. Тираж 200 экз. Заказ 675.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова, 2-й учебный корпус, 627 к.
Тел. 939-3890, 939-3891. Тел./Факс 939-3891

ISBN 978-5-89407-387-3
ISBN 978-5-317-03053-7

1. Абстрактные типы данных (АТД). Классы.

Конструкторы, деструкторы, конструктор копирования, параметры со значениями по умолчанию, работа с динамической памятью (new и delete). Ссылки, передача параметров по ссылке. Класс как область видимости. Дружественные функции. Конструкторы со списком инициализации. Статические члены класса.

1. Пусть имеется описание класса комплексных чисел

```
class complex{
  double re, im;
};
```

Написать конструкторы, позволяющие создавать объект класса complex
а) по двум заданным параметрам
б) по заданному значению re (при этом считать im=0)
в) без параметров
Можно ли написать один конструктор, удовлетворяющий всем условиям а) – в)?

2. Привести описание класса "обыкновенная дробь", члены-данные класса – числитель и знаменатель. При создании объектов этого класса возможно задание значений числителя и знаменателя или только числителя (знаменатель в этом случае считается равным 1). При создании дроби должно проводиться сокращение, то есть дроби ½ и 6/12 должны стать одинаковыми. Нужно ли для создания копий объектов такого класса явно определять конструктор копирования?

3. Описать конструктор, деструктор и конструктор копирования для класса `my_vector`

```
class my_vector{
        int len;  //количество элементов в векторе
        int *v; //указатель на массив элементов вектора
};
```

Дополнить описание класса функцией, вычисляющей длину вектора. Описать функции, вычисляющие скалярное произведение векторов, произведение вектора на константу, сумму векторов, предполагая, что оба вектора имеют одинаковую размерность.
Написать функцию-член класса `next()`, которая увеличивает на 1 все координаты вектора. Функция должна возвращать в качестве результата измененный вектор. Что произойдет с координатами вектора v после вычисления выражения `v.next().next()`?

4. Описать реализацию АТД. Если для него требуются вспомогательные АТД, привести их описание полностью.

1) **Электрический счетчик.**
   При создании объекта указывается диапазон значений, которые может отображать счетчик (при возникновении значения на 1 больше заданного максимального счетчик «сбрасывается» на 0), единицы измерения. Для счетчика должны быть определены операции увеличения значения на N и печать текущего значения.
2) **Стек для хранения целых чисел (int).**
   В классе должны быть функции добавления элемента на вершину стека (push), чтение верхнего элемента стека (pop) , проверка стека на пустоту (empty), проверка, что стек полон (full). Максимальный допустимый размер стека указывается при создании объекта.
3) **Очередь для хранения вещественных чисел (double).**
   В классе должны быть функции добавления элемента в конец очереди (put), чтение первого элемента из очереди (get) , проверка очереди на пустоту (empty), проверка, что очередь целиком заполнена (full). Максимальный допустимый размер очереди указывается при создании каждого объекта.

4) **Почтовый адрес.**

Для объектов класса должны быть предусмотрены функции изменения индекса, города, улицы, номера дома, корпуса, квартиры, вывод адреса на экран.

5) Дерево поиска.

Деревом поиска (деревом сравнений) называется двоичное дерево, которое либо пустое, либо состоит из одной вершины, либо для каждой из его вершин в левом поддереве содержатся только вершины с меньшими элементами, а в правом поддереве – только с большими элементами. Для объектов класса должны быть предусмотрены операции добавления, удаления и поиска элемента. В вершинах дерева хранятся числа типа int.

6) **Банковский счет.**

Необходимые члены-данные: дата создания счета, сумма денег, которая на нем хранится, владелец счета (объект класса person, у которого есть фамилия и имя), информация о последних 10 операциях, проведенных со счетом. Операция, проводимая со счетом – объект, содержащий дату операции, вид операции (добавить/снять деньги) и сумму операции.

У класса банковский счет должны быть методы: пополнить счет, снять деньги (если указана недопустимая сумма – должно печататься сообщение об ошибке, сумма на счете при этом не меняется), распечатать информацию о последних 10 операциях, распечатать доступную сумму денег на счете.

7) **Зоопарк.**

Необходимые члены-данные: количество зверей, массив, содержащий информацию о зверях (имя, номер клетки, название любимой еды), часы работы зоопарка адрес зоопарка, фамилия сторожа. В классе нужно определить методы, позволяющие «накормить» зверя (изменить значение соответствующего поля), поменять сторожа зоопарка (записать другую фамилию сторожа в соответствующее поле).

5. Есть ли ошибки в приведенном ниже фрагменте программы. Если да – исправить их, не изменяя функций `f`, `main` и не определяя новых функций. Перечислить все конструкторы и деструкторы, которые будут вызваны при работе программы.

```
class cat{
   double weight;
   char  name[100];
public:
   explicit cat(const char* n, double w=12.0): weight(w){
               strcpy(name,n);
            }
            cat(double w): weight (w) {
               strcpy(name,"cat");
            }
};

int f(const cat c1, const cat c2, const cat& c3,const cat & c4)
{
  return c1.weight+c2.weight+c3.weight+c4.weight;
}

int main(){
  f("Murzik", 22.0, "Kuzya",10.2);
  return 0;
}
```

6. Из указанных классов перечислить те, объекты которых можно создавать без параметров (используя конструктор умолчания).

```
class Journal{
     int pages;
     char* editor;
     int number;
public: Journal(int p, char* ed,int num):pages(p),number(num){
               editor=new char[strlen(ed)+1];
               strcpy(editor,ed);
          }
          ~Journal(){delete[] editor;}
};
class Book{
     int pages;
     char* name;
     int year;
     Book(int p=10){
          pages=p;
          name=new char [1];
          *name='\0';
          year=2007;
     }
     ~Book(){delete[] name;}
};
class Page{
     int num;
     int format;
};
class CopyBook{
     enum size {A0,A1,A2,A3,A4,A5};
     int pages;
     size sz;
     char name[20];
public:
     CopyBook(const CopyBook& c){
      sz=c.sz;
      strcpy(name,c.name);
     pages=c.pages;
     }
};
```

7. Дано описание класса Date

```
class Date{
   int day, month, year;
   static Date today;
public:
   static int count;
// В поле count подсчитывается количество созданных дат.
};
```

Написать для этого класса конструкторы (один или несколько) и деструктор. Предполагается создавать объект-дату по заданным значениям дня, дня и месяца или дня, месяца и года (недостающие значения берутся из даты `today`), корректность задания параметров в конструкторе проверять не требуется. Конструкторы и деструктор должны

учитывать появление и удаление объекта в поле `count`. Привести пример функции main, в которой продемонстрированы разные способы создания объектов-дат. Какое число будет получено в результате работы функции `f()`?

```
int f(){
    Date d1(1,1,2009), d2(28,2),d3, d4=d1;
    return d1.count + Date::count;
}
```

8. Есть ли ошибки в приведенной ниже программе на Си++? Если есть – объясните, в чем они заключаются. Если нет - прокомментировать работу программы.

```
class Flower{
     int num;
     int height;
     char* name;
public:
   Flower(int n=10, int h, char* nm = "Rose"): num(n),height(h){
          name=nm;
   }
   void ~Flower(){delete[] name;}
 };
class Garden{
     Flower& f;
public:
     void add_flower(Flower& f1){
          f=f1;
     }
};
int main(){
    Garden g;
    Flower f;
    return 0;
}
```

9. Для автоматизированной системы регистрации на курсы кройки и шитья описать классы `Course` и `Student`. Класс `Course` описывает один конкретный курс. В нем должно быть предусмотрено хранение информации об общем количестве слушателей этого курса, максимальное допустимом количестве слушателей для этого курса. У слушателя курсов (Student) должны быть определены фамилия, имя, номер паспорта, информация о курсе, на котором он учится. Предусмотреть функции добавления слушателя и исключения его с курсов. Один и тот же слушатель не может учиться на нескольких курсах. Написать функцию, не являющуюся членом указанных классов, которая для двух параметров Student проверяет, учатся ли они на одном курсе.

10. Привести примеры различных ситуаций, в которых будет вызываться конструктор копирования. Привести пример описания класса, копирование объектов которого недопустимо.

11. Привести пример класса, имеющего конструктор умолчания. Привести примеры двух различных ситуаций, в которых работает функция-конструктор (но не конструктор копирования)? Сформулируйте условия, при которых конструктор копирования по умолчанию в классе не создается.

12. Дать определение деструктора. Привести примеры двух различных ситуаций, в которых вызывается деструктор.

13. Есть ли ошибки в следующем фрагменте программы? Если да – объяснить, в чем они заключаются, и исправить их. Для каждого вызова функции внутри main, используя операцию **::** , указать, функция из какой области видимости будет вызвана.

```
class my_str{
     size_t len;
      char * str;
public:
     my_str(const char* s): len(strlen(s)){
          s = new char[len+1];
          strcpy(str,s);
     }
     ~my_str(){  delete[] str; }
     void strcpy(const my_str& t){
          if (len==t.len) strcpy(str,t.str);
          else {
               delete[] str;
               str=new char[t.len+1];
               strcpy(str,t.str);
          }
          len=t.len;
     }
};
int main(){
    my_str a("aaa"), b("bb");
     const my_str c("ccc");
     b.strcpy(c);
     strcpy(b,a);
     b.strcpy(b);
     c.strcpy(a);
     strcpy(a.str,b.str);
     return 0;
}
```

14. Дополнить программу, не изменяя текст функции main, чтобы функция main работала в соответствии со своим описанием, приведенным в комментариях. В описании классов не использовать открытые нестатические члены - данные.

```
a) int main(){
     Ball gb("green",20),//мяч цвета "green", диаметр - 20
       wb(12),//мяч цвета "white", диаметр -  12
       b(10);//мяч цвета "white", диаметр -  10
    cout<<"The smallest:"<<smallest(gb,wb,b)<<" end "<<endl;
      //должен быть напечатан диаметр самого маленького из мячей
    return 0;
   }
```

//В результате работы программы должно быть напечатано:
The smallest: 10 end
white
white
green

б)
```
int main(){
    tiger T1,T2("Kuzya");
    T1.setname("Murzik").setcolor("light").setweight(200);
             //задаем параметры тигра
    T1.print(); //печать информации о тиграх
    T2.print();
    T2.setname("Tigr");
    if (less(T2,T1)) T2.print();
      //печать информации о том тигре, который по размеру меньше
     else T1.print();
    //печатаем количество объектов-тигров
    cout<<tiger::count<<endl;
    return 0;
}
```

2. Перегрузка функций и операторов

Перегрузка функций, алгоритм выбора перегруженной функции. Перегрузка операций с помощью функций-членов класса, с помощью функций-друзей класса.

1. Пусть класс Door описан следующим образом:
```
class Door{
     int height;
     int width;
};
```
Привести примеры операций, которые
а) могут быть перегружены только с помощью функции-члена класса;
б) могут быть перегружены только с помощью функции, не являющейся членом класса;
с) могут быть перегружены как функцией - членом класса, так и функцией, не являющейся членом класса.
d) не могут быть перегружены.
Написать соответствующие прототипы перегруженных операций для класса Door.

2. Перечислить все функции, которые будут сгенерированы компилятором для следующего класса, и написать их прототипы.
```
class point {
 double x,y,z;
};
```

3. Дайте определение конструктора преобразования. Приведите пример класса, имеющего такой конструктор. Написать фрагмент программы, при выполнении которого указанный конструктор используется для преобразования типа.

4. Описать класс rational для представления действительного числа в виде обыкновенной дроби (члены-данные класса – числитель и знаменатель). Для работы с объектами класса должны быть представлены следующие функции:
а) конструктор, позволяющий строить объект по заданным значениям числителя и знаменателя или по заданному значению числителя (знаменатель в этом случае считать равным 1). Если дробь представляет собой отрицательное число, минус ставится в числителе. Если значения задают сократимую дробь, она должна быть сокращена в конструкторе. Может ли написанный конструктор использоваться как конструктор преобразования?

б) +, -, *, / для выполнения арифметических операций с дробями. Если в результате операции получается сократимая дробь, ее нужно сократить. Операции должны быть применимы для двух дробей, для первого аргумента - дроби и второго - целого числа и, наоборот, для числа и дроби.
в) операции сравнения рациональных чисел
г) оператор приведения к типу double
Объясните, нужно ли для данного класса определять деструктор, конструктор копирования, операцию присваивания?

5. Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются и вычеркните их из текста функции f(). Что будет напечатано при вызове получившейся функции f()?

```
class A {
    int a;
    A(A &b) { a = b.a; cout <<1;}
 public:  A(int a = 0) { this -> a = a; cout <<2;}
          A( ) { a = 100; cout <<3;}
          ~A( ) { cout << 4;}
         void operator= (const A &b) {a = b.a; cout << 5; }
};

void f() {
  A x(1);  A y;  A z = A(2);  A t(3.8);  A w('5');
  x = w = t;
}
```

6. Описать класс my_vector для работы с векторами n-мерного пространства. Определить для векторов операции:
а) присваивание (=),
б) сложение (+) – координаты вектора результата равны сумме соответствующих координат векторов-слагаемых,
в) сложение с присваиванием (+=),
г) умножение (*) на число слева и справа,
д) скалярное произведение(^),
е) индексирование ([ ]), допускающее использование результата как слева, так и справа от присваивания.

7. Для заданного класса stack перегрузить оператор ! всеми возможными способами (член класса, внешняя по отношению к классу функция). Оператор в качестве результата возвращает стек, который является копией исходного, но у всех его элементов знак изменен на противоположный.

```
class stack{
  int * s; //указатель на хранилище элементов стека
  int top; //номер последней занятой позиции
  int size;//размер стека
public:
  stack(int max=100) {
   top=-1;size=max; s=new int[max];
   };
 ~stack(){delete [] s;}
};
```

8. Нужно ли определять оператор = для класса `stack` из задачи 7? Если да - привести пример программы, которая будет работать некорректно, если пользовательский оператор = не будет написан. Если нет – объяснить, как будет выполняться присваивание для объектов `stack`.

9. Сформулируйте алгоритм выбора перегруженной функции.
Пусть в программе имеются следующие описания классов и функций:

```
class A {
 int m;
 friend class B;
public: A (int n):m(n){}
};

class B {
 int p,q;
 friend class C;
public: B(){p=q=0;}
        B(const A& a){p = a.m; q = a.m;}
};

class C{
 int c1,c2,c3;
public: C(const B& b):c1(b.p),c2(b.q),c3(b.q){}
};

void print(const char *s){cout<<" 1 "<<s<<endl;}
void print(int d){cout<<" 2 "<<d<<endl;}
void print(double d){cout<<"3"<<d<<endl;}

void print1(const A&){cout<<"Object A"<<endl;}
void print1(const C&,int t){cout<<"Object C"<<endl;}

void print2(int a, int b){cout<<"print 2 int"<<endl;}
void print2(double a, double b){cout<<"print 2 double"<<endl;}
void print2(int a,...){cout<< "print 2..."<<endl;}
```

Допустимы ли следующие вызовы функций? Если да, то прокомментировать, какие функции будут вызваны. Если нет – объяснить, в чем заключается ошибка

```
print(1);
print(12UL);
print(true);
print("test");
print(0.0);
print(0.0f);

print1(12.9);
print1(11,4);
B b; print1(b,2);

print2(14,14);
print2('1',14);
print2(1.0,14);
print2(1,2,3);
```

10. Описать класс my_str, содержащий длину строки и указатель на ее первый символ. Определить для класса необходимые конструктор(ы), деструктор, а также операции
а) присваивания (=), осуществляющее посимвольное копирование строк;
б) сложения (+) – результатом является конкатенация исходных строк;
в) присваивание со сложением +=, к первой строке-параметру добавляется в конец вторая строка;
г) приведения к типу char*;
д) ввод (>>) строки, который осуществляется до символа '\n', все введенные символы-пробелы пропускаются (не добавляются в строку).

11. Для заданного класса point перегрузить оператор *= всеми возможными способами (член класса, друг класса). Оператор *= должен быть применим для двух объектов класса point, он осуществляет умножение соответствующих координат точек и запись полученных результатов в координаты первой точки.

```
class point{
   int *x,*y,*z;//указатели на координаты точки
public:
   point(int c1=0, int c2=0, int c3=0)   {
     x=new int(c1); y=new int(c2); z=new int(c3);
   }
  ~point(){
     delete x;  delete y; delete z;
   }
};
```

12. Будет ли для класса point из задачи 11 по умолчанию сгенерирована операция == (сравнение на равенство)? Если да, объяснить, как будут выполняться сравнения на равенство для объектов point. Если нет, перегрузить эту операцию так, чтобы она осуществляла сравнение соответствующих координат точек.

13. Для класса Date, описывающего дату (день, месяц год), перегрузить оператор вывода, позволяющий печатать даты в формате 01 Jan 2009, и оператор ввода, позволяющий вводить даты как три последовательности десятичных цифр, разделенных произвольным количеством пробелов. Например, 01 01 2009 - это дата 1 января 2009 года.

14. Привести описание класса fridge и, если необходимо, дополнительные функции, чтобы следующая функция main работала в соответствии со своим описанием, приведенным в комментариях.

```
int main(){
  fridge R("blue",7,1,12); //цвет, длина, ширина, высота
  fridge G( 7);//холодильник белого цвета 7x7x7
  fridge Res = R++; /* оператор ++ увеличивает на 1 все размеры
                       Res – копия R до увеличения размеров*/
  cout<<Res<<R<<G;//вывод информации о холодильниках
  Res =G+R;  /*цвет – любой, каждый размер = сумме размеров
               холодильников-слагаемых */
  Res=4+G;//та же функция сложения, что и в предыдущем операторе
  cout<<Res<<R<<G;//вывод информации о холодильниках
  Res[1]=Res[2]+10;//длина холодильника = ширина холодильника+10
  cout<<Res<<R<<++G;//вывод информации о холодильниках
  return 0;
 }
```

3. Наследование. Виртуальные функции

Квалификатор protected, перекрытие имен и перегрузка функций при наследовании. Работа конструкторов и деструкторов при создании объектов производных классов. Виртуальные функции, виртуальный деструктор. Абстрактные классы.

1. Описать иерархию классов для хранения информации о сотрудниках, студентах и аспирантах университета. В качестве базового использовать класс Person.

2. Перечислить все функции, которые могут быть вызваны из функции main для объекта класса rabbit. Какие из функций будут доступны для объекта класса rabbit в функции main, если класс будет унаследован от базового со спецификатором protected вместо public?

```
class animal{
   bool fur;
   void set_fur(){fur=true;}
 protected:
     int weight;
     void set_weight(int w){weight=w;}
 public:
     int luck;
     void set_luck(int l){luck=l;}
     bool is_lucky(){return (luck>15000);}
};

class rabbit:public animal{
 protected:
     bool hungry;
 public:
     void set_luck(double d){luck = d*1000;}
     void set_luck(){luck=10000;}
     bool is_hungry(){return hungry;}
};
```

3. Определить иерархию классов для описания сессии в университете. Базовый класс – `event`, содержит информацию о дате события, фамилию действующего лица и чисто виртуальную функцию `print_res`, печатающую информацию о событии; классы-наследники – `test` (зачет), `exam` (экзамен). В них должны быть определены дополнительные характеристики событий и метод `print_res`. В функции `main` определить сессию как массив указателей на события, проинициализировать элементы массива и распечатать информацию об экзаменах и зачетах.

4. Что будет напечатано в результате работы следующей программы?

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

class Transport {
  char num[7]; //регистрационный номер
 public:
  Transport (const char* x="B123MK") {
    strcpy(num,x);
    cout<<"T constr\n";
  }
```

```
  virtual ~ Transport(){
    cout<<"T destr\n";
  }
  virtual int h(){return 0;}
  int g(){return -1;}
};

class Bus : public Transport {
  int n;     //номер маршрута
  public:
    Bus(char* x,int nm):Transport(x),n(nm){cout<<"B constr\n";}
    Bus(const Bus& b) { cout<<"B copy\n";}
    ~Bus() { cout<<" B destr\n";}
   int h() {return n;}
   int g() {return -n;}
};

void f(Transport * t1, Transport& t2, Bus t3){
  cout<<t1->h()<<" and "<<t1->g()<<endl;
  cout<<t2.h()<<" and "<<t2.g()<<endl;
  cout<<t3.h()<<" and "<<t3.g()<<endl;
}

int main () {
 Bus b ("B123AA",130);
  f(&b,b,b);
  cout<<"end"<<endl;
 return 0;
}
```

5. Привести пример программы, которая работает по-разному в зависимости от того, был ли объявлен деструктор виртуальным или нет.

6. Найти ошибки в программе и объяснить, в чем они заключаются.

```
class table{
  int size;
  int priority;
 public:
  table(int s=0,int p):size(s),priority(p){ }
  virtual void print()=0;
};

class stud_table: public table{
     char * name;
     int gr;
public:
     void print(){cout<<"students table"<<endl;}
     ~stud_table(){delete []name;}
};

class asp_table:protected table{
    char* thesis;
};
```

```
int main(){
  table t;
  stud_table st;
  table * tp = &st;
      tp = new asp_table();
  stud_table * stp = &st;
  cout<<"Program"<<endl;
  return 0;
}
```

7. Есть ли ошибки в приведенном ниже фрагменте программы на языке Си++? Если есть, то объясните, в чем они состоят, если нет, прокомментируйте, что будет выдано в стандартный канал вывода в результате вызова функции main() ?

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

class Vehicle {
protected:
     int speed;
public:
     virtual void repair (int j) {
          cout << "Vehicle "<<j<<endl;
          if (repair()) speed+=j;
     }
     int repair(){cout << "Vehicle repair "<< endl; return 1;}
};

class Car : public Vehicle {
protected:
     char mark[20];
public :
     void repair(int j){cout << "Car " << j << endl; repair();}
     int repair () { cout << "Car repair " << endl; return 0;}
};

class CityCar : public Car {
   int seats;
public:
     void repair(int j){cout<<"CityCar"<<j<<endl;Car::repair();}
     int repair (){cout << "CityCar repair" << endl; return 0;}
};

void service (Vehicle * p1, Car * p2) {
     p1 -> repair (1); p2 -> repair (2);
}

int main () {
     Vehicle v; Car c; CityCar cc;
     Vehicle * vp; Car * cp;
     vp = &v; cp = & c; service(vp, cp);
     vp = cp; cp = &cc; service(vp, cp);
    return 0;
}
```

8. В программе присутствуют описания двух классов

```
class Animal{
  char name[20];
public:
  void print(){}
};

class Elephant: protected Animal{
  int weight;//вес
  int trunc_len;//длина хобота
public:
  void print(){cout<<"Elephant"<<endl;}
};
```

Не добавляя новых функций-членов, изменить описание этих классов, чтобы выполнялись оба условия:

а) Базовый класс является абстрактным

б) Допустим следующий фрагмент программы:

```
 int main() {
    const Elephant s;
    const Animal * a = &s;
    a->print();
    return 0;
}
```

4. Обработка исключений

Генерация исключений при возникновении ошибочных ситуаций, обработка исключений, алгоритм поиска подходящего обработчика. Генерация новых типов исключений на основе исключений стандартной библиотеки.

1. Описать класс Date, для работы с датами (содержащий информацию о дне, месяце и годе). В классе должна быть описана функция set_date, которая устанавливает значения дня, месяца и года для даты, а при задании некорректных параметров генерирует исключение. Написать в функции main примеры вызовов написанной функции для демонстрации работы с корректными и с некорректными значениями. При возникновении исключения – печатать сообщение об ошибке.

2. Что такое свертка (раскрутка) стека? Что будет, если во время свертки стека в деструкторе возникает исключение?

3. Переопределить функции terminate и unexpected с помощью set_terminate и set_unexpected, чтобы функции печатали свое имя, а затем завершали работу программы. Продемонстрировать на примере работу переопределенных функций.

4. Промоделировать конструкцию void f() throw(T1,T2,T3), считая, что компилятор поддерживает throw, throw выражение, catch, но не поддерживает throw(список типов).

5. Что будет напечатано в результате работы программы

а)

```
class animal{
     int weight;
   public:
     ~animal(){cout<<"animal destructor"<<endl;}
  };
```

```
class elephant:public animal{
  int trunk_length;
  char * color;
public:
     ~elephant(){cout<<"elephant"<<endl;}
};

int main(){
  elephant jumbo;
  animal* a = new elephant();
  try{
     try{
      elephant kuzia;
      throw "error";
      cout<<"message 1"<<endl;
     }
     catch(int ){cout<<"catch 1"<<endl;}
     catch(const void* ){cout<<"catch 2"<<endl;throw;}
     catch(const char* s){cout<<s<<"catch 3"<<endl;}
  }
  catch(const char* s){cout<<s<<" catch 4"<<endl;}
  catch(...){cout<<"All exceptions"<<endl;}
  delete a;
  return 0;
}
```

б)
```
int f(int k) {
 try { k++;
     switch (k) {
          case '1': throw 2;
          case 1  : throw 1;
          case 0  : throw "Exception";
     };
     return 100;
 }
 catch (int k){ cout << k << " catchl\n"; throw;}
 catch (const char*){ cout << " catch2\n"; return 50;}
}

class Box {
     int d;
public:
     Box(int j) { d = f(j); cout << d <<endl;}
};

int main(){
  try {
   Box b(-1),c(0),a(12); cout<<"Finish"<<endl;
  }
  catch (int){ cout << "catch3\n";}
  catch (const char *){ cout << "catch4\n";}
  return 0;
}
```

в)
```
class X {
  int j;
public:
  X(int k){
    try{
      if (!(k/10)) throw k;
      switch (k){
      case 5: j=0;
      case 6: j=1;
      default: j = ++k;
      };
      cout<< j <<endl;
   }
   catch(double){cout<<" catch0 "<<endl;}
  }
};

void test(int i) {
    try {
      X a(i), b(i-14);
    }
    catch(int n){cout<<n<<" catch1 "<<endl; if(n>5) throw;}
    catch (...) {cout<<" catch2 "<<endl;throw;}
}

int main (){
     try {
       test(1); test(20); test (85);
       cout<<"try 1"<<endl;
     }
     catch (int) {cout<<"catch2"<<endl;}
     catch (...) {cout<<"catch3"<<endl;}
     return 0;
}
```

6. Описать класс MagicFlower. Для создания объектов этого класса необходимо задать число лепестков цветка, которое должно быть совершенным числом, и цвет лепестков (white или pink). Совершенным называется число, равное сумме всех своих делителей. При задании некорректного числа лепестков цветка должно возникать исключение. Если задан неверный цвет, создать цветок такого цвета, объектов которого было создано меньше.

7. Шахматы. Определить класс "шахматная доска", описывающий расположение фигур на шахматной доске. В конструкторе класса предусмотреть проверку корректности расстановки фигур (например, черный слон не может находиться на белой клетке). Среди методов класса должен присутствовать метод step, моделирующий движение фигуры по шахматной доске. В качестве параметров метод step получает название фигуры, ее цвет и клетку, в которую фигура должна перейти. Если переход в указанную клетку по какой-либо причине невозможен - сгенерировать исключение типа const char*, в котором передать сообщение о возникшей ситуации. Продемонстрировать работу с классом.

8. Описать необходимые классы и объекты, моделирующие заданную ситуацию. Все исключения, генерируемые пользователем должны быть наследниками класса exception (стандартного исключения)

а)Тетушка Агата купила стиральную машину. Описать необходимые классы и объекты, моделирующие процесс стирки, описанный следующим образом:

```
try{
 mashine.wash();
}
catch(exception& ex){cerr<<ex.what()<<endl;}
```

Функция wash случайным образом может сгенерировать исключение одного из типов: `WaterFinished`, `DoorOpened`, `WrongObjectInside` либо же может завершить работу успешно. В обработчике `what()` должна печатать информацию о том исключении, которое возникло в процессе работы `wash`.

b) Движение на железной дороге. Описать необходимые классы и объекты для моделирования движения поезда в Африке

```
try{
 train1.go();
}
catch(exception& ex){cerr<<ex.what()<<endl;}
```

Функция go может успешно завершиться, а может сгенерировать случайным образом исключение одного из следующих типов: `WrongWay` (не переведена стрелка), `PowerFailure` (сбой электричества), `Elephants` (слоны переходят дорогу), `Stop` (нажат стоп-кран). Если в процессе движения поезда возникает исключение, он останавливается (функция go завершает свою работу), а обработчик исключения печатает, используя функцию `what()` информацию о причине остановки.

9. Написать программу, читающую последовательности символов со стандартного ввода. Признаком конца последовательности считать символ '$' (он в последовательность не входит). Обработать последовательность согласно одному из правил а) – е). При вводе пустой последовательности должно возникать исключение типа `NullException`. Типы исключений описать как классы-наследники `exception`, определить для них функцию `what()`. В функции `main` должна быть предусмотрена обработка возникших исключений. Информацию о типе исключения выдавать на экран.

а) Если введенная последовательность символов является записью целого числа в шестнадцатеричной системе счисления – напечатать это число в десятичном виде и тот целый тип, в котором это число можно разместить. Если введенное число не помещается ни в один из встроенных целых типов – выдать исключение типа `OutOfRange`. При вводе строки, не удовлетворяющей условию задачи – генерировать исключение типа `IllegalNumber`.

б) Распечатать последовательность символов, полученную из исходной удалением всех вхождений цепочки символов C++. Если цепочка C++ ни разу не встретилась в исходной последовательности – сгенерировать исключение типа `InvalidInput`.

в) Распечатать сумму всех целых десятичных чисел, которые встретились в строке. Если эта сумма не помещается в тип int – сгенерировать исключение типа `OverflowException`. Если не встретилось ни одного числа – исключение типа `NoDigits`.

г) Распечатать для всех символов последовательности сколько раз они в ней встретились (если символ в строке не встретился – о нем ничего не печатать). Если не встретилось ни одной строчной латинской буквы – выдать исключение типа `NoLetterException`.

д) Все группы стоящих подряд пробелов заменить одним символом пробел. Если строка состояла только из пробелов – выдать исключение типа `OnlySpacesException`.
е) Удалить из каждой группы подряд идущих цифр все незначащие нули (если группа состоит только из нулей – заменить эту группу одним нулем). Если в последовательности нет ни одной цифры - выдать исключение типа `NoDigitException`.

10. Пусть в программе присутствуют следующие описания классов.

```
class Base {
 protected:
    int b1,b2
public:
    virtual void print()=0;
};

class Derived: public Base{
  int d;
public:
   void print(){cout<<b1<<b2<<d<<endl;}
};
```

Заменить в функции main операторы приведения типов в стиле Си на подходящие операторы приведения типов в стиле Си++. Приведения типов, которые могут быть выполнены неявно, вычеркнуть.

```
int main(){
  void * vp;
  const char * str = "test string";
  int x=5,y=10;
  Base * bp;
  Derived d, *dp;
    vp = str;
    cout<<x /(double) y << (const char*) vp <<endl;
    bp = (Base *) &d;
    dp = (Derived *) bp;
  return 0;
}
```

11. В приведенной ниже функции main написать преобразования типов так, чтобы все вызовы функций были корректными. Использовать преобразование const_cast (там, где оно не может быть выполнено неявно).

```
 void f1(const int & x){cout<< one << x <<endl;}
 void f2(int & x){cout<< two << x <<endl;}
 void f3(const int  x){cout<< three << x <<endl;}
 void f4(int  x){cout<< four << x <<endl;}
 void f5(const int * x){cout<< five << *x <<endl;}
 void f6(int * x){cout<< six << *x <<endl;}

 int main(){
  const int c = 8;
  int i = 45;
  f1(c); f2(c); f3(c); f4(c); f5(&c); f6(&c);
  f1(i); f2(i); f3(i); f4(i); f5(&i); f6(&i);
  return 0;
 }
```

5. Шаблоны

Создание шаблонов функций и шаблонов классов. Алгоритм выбора перегруженной функции с учетом шаблонных функций.

1. Описать шаблонную функцию max, которая возвращает значение максимального элемента для заданного массива целых чисел (int), длинных целых чисел (long), массива вещественных чисел (double) или массива строк (const char*). При работе со строками, максимальной считать ту строку, которая имеет наибольшую длину.

2. Описать шаблонную функцию copy, которая копирует значения элементов второго параметра-массива в первый параметр-массив целых чисел (int), длинных целых чисел (long), вещественных чисел (double) или строк (char*)

3. Сформулируйте алгоритм выбора перегруженной функции с учетом шаблонов.

4. Пусть описана шаблонная функция

```
template <class T>  T max (T& x, T& y) {return x > y ? x : y;}
```

Прокомментируйте выполнение приведенного ниже фрагмента программы (считая, что кроме шаблонной функции max других функций с таким именем нет).

```
double x = 1.5, y = 2.8, z;
int i = 5, j = 12, k;
char *s1 = "mouse";
char *s2 = "house", *s3;
z = max(x, y);
k = max <int>(i, j);
z = max(x, i);
z = max <double>(y, j);
s3 = max(s1, s2);
```

5. Описать шаблонный класс Set для представления множества. Для объектов класса Set определить операции добавления элемента в множество (add), удаление элемента (del), проверка вхождения элемента (in), пересечение (*), объединение(+) и разность (-) множеств.

6. Описать шаблонный класс List для работы с однонаправленными списками. Для объектов класса List определить операции проверки списка на пустоту, добавления элемента в начало списка, в конец списка, подсчет числа вхождений элемента в список, удаление элемента из списка. Продемонстрировать работу с шаблонным классом для списка с целыми элементами и с элементами-строками.

6. Практические задания

Описанное в условиях задач взаимодействие объектов требуется промоделировать средствами языка C++. Выполнение задач этого раздела, включающих параллельное выполнение процессов, обработку сигналов предполагается в ОС UNIX.

1. C и C++

Среди начинающих программистов бытует мнение, что язык C++ является расширением языка C. Более строго этот тезис сформулировать можно так: ”Если программа работает на C, то она будет работать и на C++”. Однако, это неверно. Приведите несколько различных примеров программ, которые успешно компилируются и работают как программы на C, но либо не компилируются, либо работают иначе, будучи рассмотренными как программы на C++.

2. Арифметика длинных чисел

Определить класс, реализующий арифметические операции (+, - , * , / , %) для работы с целыми числами произвольной длины. Создать объекты класса и продемонстрировать работу написанных операций.

3. Предметный указатель

Определить класс для работы с предметными указателями. Предметный указатель строится по заданному текстовому файлу. Словом в таком файле является последовательность непробельных символов, разделители между словами — пробел, табуляция и символ перевода строки. Каждый компонент указателя — содержит слово и номера строк, на которых это слово встречается (количество вхождений слова в файл заранее не известно).

4. Разработка иерархии классов

Выбрать предметную область, в которой можно выделить объекты как минимум трёх типов, связанные между собой иерархическими отношениями. Используя наследование, описать классы для работы такими объектами. Дополнительные условия:

1. Базовым должен являться абстрактный класс. Среди его членов-данных должен присутствовать указатель.
2. В программе должны присутствовать операции, перегруженные как члены класса и операции, перегруженные с помощью функций, не являющихся членами класса.
3. В программе должны создаваться объекты разработанных классов, они должны взаимодействовать, используя описанные методы и перегруженные операции.
4. В описании классов должны быть описаны как минимум две виртуальные функции, работа которых также должна быть продемонстрирована в программе.

5. Игра в города[1]

Всем известны правила игры в города: первый игрок называет произвольный город, следующий – город, название которого начинается на ту же букву, на которую закончилось название предыдущего города и т.д.

Задан список допустимых для описанной игры городов (слова в нем не повторяются). Разработать необходимые АТД и использовать их для решения следующей задачи: определить, в каком порядке в процессе игры должны быть названы города. В игре должны быть использованы все названия городов из списка, а если такую цепочку слов построить не удалось, соответствующая функция должна выдавать исключение.

Входные данные:
В первой строке входного файла записано целое число N – количество слов в списке (1<=N<=1000), а в последующих строках – сами слова. каждое из них - последовательность не более чем из 10 строчных латинских букв. Для работы с файлом использовать библиотеку fstream.h (или fstream).
Выходные данные:
Названия городов в искомом порядке (если возможно несколько вариантов упорядочения – напечатать один из них, любой) или диагностическое сообщение, если при построении списка возникло исключение.
Пример входного файла:
4
Kemerovo
Minsk

[1] оригинал задачи описан в [8]

Astana
Odessa
Пример выходного файла:
Minsk
Kemerovo
Odessa
Astana
Комментарий: Одним из возможных подходов к решению является следующий: нужно составить ориентированный граф, вершинами которого являются буквы от a до z. Каждому слову из списка сопоставим ребро, соединяющее первую и последнюю буквы слова. Одна и та же пара букв может быть соединена несколькими ребрами. В полученном графе требуется найти Эйлеров путь, то есть путь, проходящий по каждому ребру ровно один раз.

6. Великий комбинатор[2]

В середине 70-х годов в Англии широкую популярность получила игра "Великий комбинатор" (Mastermind), известная у нас под названием "Быки и коровы". Правила игры следующие. **Ведущий** загадывает комбинацию из 4 цифр (от 1 до 6, цифры могут повторяться), называемую кодом. **Игрок** пытается раскрыть код, высказывая разумные предположения, называемые пробами. Каждая проба, как и код, представляет собой четверку цифр. Ведущий дает ответ о количестве цифр, которые есть в коде и их позиции указаны верно (они называются быками), и о количестве цифр, которые присутствуют в коде, но стоят на других позициях (коровы). Игра продолжается до тех пор, пока число не будет угадано. Пример игры:
загадано число 1234
проба 1: 4560 ответ : быков – 0, коров – 1.
проба 2: 1143 ответ : быков – 1, коров – 2.
проба 3: 1234 ответ : быков – 4, коров – 0. Число угадано.
Написать программу, моделирующую игру "Быки и коровы". Для этого описать необходимые АТД и продумать взаимодействие объектов этих типов.
Ведущий загадывает код из 4-х цифр, сгенерированный случайным образом. В игре присутствуют как минимум два игрока.
Первый игрок считывает пробы со стандартного ввода, печатает ответы ведущего на стандартный вывод. Ввод пробы $$$$ является отказом от игры.
Второй игрок-компьютер. Для него нужно разработать и реализовать алгоритм генерации проб. Информация о ходе его игры (пробы и ответы на них) печатается в файл, имя которого задается в командной строке при запуске программы.
При нажатии Ctrl+C программа должна напечатать текущее состояние игры: закончена она или нет, сколько проб сделал каждый из игроков.
По окончании игры ведущий должен сообщить, за сколько ходов число было угадано каждым из игроков. На этом программа завершается.

7. Белки в космосе.

Когда астрономы нашли новую планету, на космическом корабле туда была отправлена белка. Белке очень понравилось новое место обитания, и через месяц белок на планете стало двое. Далее белки продолжали плодиться с той же скоростью, то есть каждый месяц на планете каждая белка производила на свет еще одну белку. Однако неожиданно на планете обнаружился космический тигр. Как только белок становилось K или больше, он вылезал из своей пещеры и съедал ровно K белок.

[2] оригинал задачи и возможные подходы к решению описаны в [9]

Написать программу, моделирующую заселение разных планет. Названия планет и для каждой из них число К - "предел терпения тигра" задаются в командной строке. Для каждой планеты создается отдельный объект, а заселение планеты контролируется отдельным процессом. Считать, что месяц проходит за 10 секунд.
При запуске основного процесса пользователю предлагается ввести название планеты, чтобы получить информацию о живущих там белках. Если введено название существующей планеты, печатается информация о том, сколько белок живет в этот момент на указанной планете. Если название планеты введено неверно - выдается диагностическое сообщение и предлагается ввести название планеты снова и т.д. При нажатии комбинации клавиш Ctrl+C должен быть напечатан список названий планет, где белок уже не осталось, а программа продолжает выполняться дальше.
ПРИМЕР РАБОТЫ ПРОГРАММЫ
В командной строке заданы названия двух планет и пределы терпения тигра:
./squirrels.out Марс 8 Луна 2
Послав запрос для Марса и Луны через 10 секунд, узнаем, что на Марсе будет 2 белки, на Луне - 0 (так как только что появившиеся две белки были съедены тигром)
Еще через 12 секунд - на Марсе будет 4 белки. При нажатии Ctrl+C будет напечатано, что на Луне белок нет, затем программа продолжит ждать ввода имени планеты, информация о которой интересует пользователя.

Литература

1. Страуструп Б. "Язык программирования C++" Страуструп Б.. Язык программирования C++. Специальное издание, М., Бином, 2005.
2. Мейерс С. Эффективное использование C++. 50 рекомендаций по улучшению ваших программ и проектов. Питер, ДМК Пресс, Москва, 2006
3. Мейерс С. Эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов. Питер, ДМК Пресс, Москва, 2006
4. Пол А. Объектно-ориентированное программирование на C++. Второе издание, М, Бином, СПб, Невский диалект, 1999.
5. Руденко Т.В. "Сборник задач и упражнений по языку Си" М., Издательский отдел факультета ВМиК МГУ, 1999.
6. Робачевский А.М., Немнюгин С.А., Стесик О.Л. Операционная система UNIX. Второе издание, СПб, БХВ-Петербург, 2005.
7. Вдовикина Н.В., Казунин А.В., Машечкин И.В., Терехин А.Н. Системное программное обеспечение. Взаимодействие процессов. Учебно-методическое пособие. М., Издательский отдел факультета ВМиК МГУ, 2002.
8. Беров В.И., Лапунов А.В., Матюхин В.А., Пономарев А.Е. Особенности национальных задач по информатике. Киров, Триада-С, 2000
9. Уэзерелл Ч. Этюды для программистов, М., Мир, 1982
10. Стандарт C++ ISO / IEC 14882 : 1998(E)

СОДЕРЖАНИЕ