Дополнительный список задач по Си++

Задача №1

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются.

```
class A
{
  int a, b;
public:
  A ( A &x )  { a = x.a; b = x.b; cout << 1; }
  A ( int a ) { this->a = a; b = a; cout << 2; }
};

void f ()
{
  A x ( 1 );
  A y;
  A z = A ( 2, 4 );
  A t ( 3.5, 2 );
  A s = 6;
  A w = z;
  t = z = w;
}
```

Как нужно изменить описание класса *A* (не изменяя описания функции *f*), чтобы в *f*() не было ошибок? Что будет напечатано в результате работы функции *f*()?

Обязательное требование: нельзя добавлять новые функции-члены класса *A*, но допустимы любые изменения в уже имеющихся.

Задача №2

Написать конструктор для некоторого класса A таким образом, чтобы были выполнены следующие условия:

- это единственный явно описанный конструктор класса A,
- верны следующие описания объектов класса A:

```
A a;
A b ( 1 );
A c ( 1, 2 );
A d ( '1', 1 );
```

Задача №3

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Вычеркните ошибочные конструкции (если они есть). Что будет напечатано при вызове функции f()?

```
class A
{
  int a;
public:
  A ( int x )                   { a = x; cout << 1; }
  A ( int x, int y )            { a = x / y; cout << 2; }
  A ( int x, int y, int z = 1 ) { a = (x+y) / z; cout << 3; }
 ~A ()                          { cout << 4; }
  void operator = ( A &b )      { a = b.a; cout << 5; }
};

void f ()
{
  A x;
```

```
  A y ( 1 );
  A z ( 1, 2 );
  A t ( 1, 2, 3 );
  A w = 4;
  A v = y;
  y = t = v;
}
```

Задача №4

Для класса string опишите конструктор копирования и операцию присваивания с объявленными профилями, реализующие «глубокое» («содержательное») копирование объектов этого класса. Объясните, почему был выбран именно такой вариант объявления этих функций-членов класса.

```
class string
{
  char *p;
  int size;
public:
  string ( const char *str )
  {
    p = new char [(size = strlen ( str )) + 1];
    strcpy ( p, str );
  }
 ~string () { delete[]p; }
  string ( const string &a );
  string& operator= ( const string &a );
};
```

Задача №5.

Для класса *rational* опишите перегруженную операцию умножения так, чтобы были верными все действия функции g(). Возможно ли это? Если нет — объясните почему; если можно - опишите и объясните, какие действия будут выполняться при вызове функции g()? Если возможно несколько вариантов решения этой задачи, то укажите их все.

```
class rational
{
  int p, q;
public:
  rational ( int a , int b )
  {
    p = a;
    q = b;
  }
  // ...
};

void g ()
{
  rational x ( 1, 2), y ( 4, 5 ), z ( 1, 1 );
  const rational w ( 1, 8 );
  int t = 3;
  z = x * y;
  z = w * y;
  z = y * w;
  z = z * t;
  z = t * z;
}
```

Задача №6

Для класса *rational* опишите перегруженную операцию инкремента (префиксный и постфиксный варианты) так, чтобы были верными все действия функции *g*(). Семантика этой операции — увеличение на 1 значений полей *p* и *q*. Возможно ли это? Если нет — объясните почему; если можно — опишите и объясните, какие действия будут выполняться при вызове функции *g*()? Если возможно несколько вариантов решения этой задачи, то укажите их все.

```
class rational
{
  int p, q;
public:
  rational ( int a, int b )
  {
    p = a;
    q = b;
  }
  // ...
};

void g ()
{
  rational x ( 1, 2 ), z ( 1, 1 );
  const rational y ( 1, 8 );
  z = x++;
  z = y++;
  z = ++x;
  z = ++y;
}
```

Возможно ли для объектов класса *rational* с перегруженной вами операцией инкремента следующее:

```
z = x++ ++;
z = ++ ++x;
```

Если возможно, то опишите последовательность действий при выполнении этих операторов и укажите, чему равны значения x и z. Если невозможно, то объясните почему.

Задача №7

Для класса string опишите перегруженную операцию конкатенации так, чтобы были верными все действия функции g(). Возможно ли это? Если нет — объясните почему; если можно — опишите и объясните, какие действия будут выполняться при вызове функции g()? Если возможно несколько вариантов решения этой задачи, то укажите их все.

```
class string

{
  char *p;
  int   size;
public:
  string ( const char *str)
  {
    p = new char [(size = strlen(str)) + 1];
    strcpy( p, str );
  }
  string ( const string &a )
  {
    p = new char [(size = a.size) + 1];
    strcpy ( p, a.p );
  }
 ~string ()
  {
    delete[] p;
  }
  string& operator= ( const string &a )
  {
    if ( this == &a ) return *this;
    delete [] p;
    p = new char [(size = a.size) + 1];
    strcpy ( p, a.p );
    return *this;
  }
};

void g ()
{
  string s1 ( "11111" );
  string s2 = s1;
  string s3 ( "222222222" );
```

```
  const string s4 ( "4444444" );
  // ...
  s1 = s2 + s3;
  s1 = s3 + s4;
  s1 = s4 + s3;
  s1 = s3 + "abcd";
}
```

Задача №8

Для класса *string* опишите перегруженную операцию * так, чтобы были верными все действия функции g(). Результат умножения строки s на целое число n (и целого числа n на строку s) — это строка, полученная в результате конкатенации n строк s. Возможно ли это? Если нет — объясните почему; если можно — опишите и объясните, какие действия будут выполняться при вызове функции g()? Если возможно несколько вариантов решения этой задачи, то укажите их все.

```
class string
{
  char *p;
  int   size;
public:
  string ( const char *str )
  {
    p = new char [(size = strlen(str)) + 1];
    strcpy ( p, str );
  }
  string ( const string &a )
  {
    p = new char [(size = a.size) + 1];
    strcpy ( p, a.p );
  }
 ~string () { delete[] p; }
  string& operator= ( const string &a )
  {
    if ( this == &a ) return *this;
    delete [] p;
    p = new char [(size = a.size) + 1];
    strcpy ( p, a.p );
    return *this;
  }
};

void g ()
{
  string s1 ( "11111" );
  string s2 = s1;
  const string s3 ( "222222222" );
  int t = 5;

  // ...
  s1 = s2 * t;
  s1 = t * s2;
  s1 = s3 * t;
  s1 = t * s3;
  s1 = s2 * 3;
}
```

Задача №9

Что будет выдано в стандартный канал вывода при вызове функции *main*()? Объясните ответ.

```
void f ( char c )
{
  try
  {
    if ( c >='1' && c<='9')
      c++;
    else
      throw "error";
    try
    {
      if ( c == '8' )
        throw c;
```

```
      c++;
    }
    catch ( const char *c )
    {
      cout << c << "\ncatch1\n";
    }
    catch ( char c )
    {
      cout << "catch2\n";
      throw;
    }
    cout << c << endl;
  }
  catch ( int k )
  {
    cout << k << "\ncatch3\n";
  }
  catch ( char *c )
  {
    cout << c << "\ncatch4\n";
  }
}

int main ()
{
  try
  {
    f ( 'a' );
    f ( '2' );
    f ( '7' );
    f ( '5' );
  }
  catch ( char c )
  {
    cout << c << "\ncatch5\n";
  }
  catch ( ... )
  {
    cout << "all exceptions\n";
  };
  return 0;
}
```

Задача №10

Что будет выдано в стандартный канал вывода при вызове функции *main*()? Объясните ответ.

```
class X
{
  int j;
public:
  X ( int k )
  {
    try
    {
      if ( k == 10 ) throw k;
      if ( k == 0 )  throw '0';
      j = k;
    }
    catch ( char )
    {
      cout << "catch1\n";
    }
    catch (...)
    {
      cout << "catch2\n";
      throw;
    }
  }
};

void f ( int i )
{
  try
  {
    X a ( i );
    X b ( 5 );
  }
```

```
  catch ( int )
  {
    cout << "catch3\n";
    throw;
  }
  catch (...)
  {
    cout << "catch4\n";
  }
}

int main ()
{
  try
  {
    f ( 0 );
    f ( 10 );
    f ( 100 );
  }
  catch ( int )
  {
    cout << "catch5\n";
  };
  return 0;
}
```

Задача №11

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Ошибочные конструкции вычеркнуть из текста программы. Какие функции будут вызваны при вызове функции g()?

```
class A
{
public:
          void f1 ()               { cout  << "f1_A" << endl; }
  virtual void f1 ( const int c )  { cout  << "f1_A_int" << endl; }
  virtual void f2 ()               { cout  << "f2_A" << endl; }
  virtual int  f3 ( const int k )  { cout  << "f3_A" << endl; return k + 10; }
};

class D: public A
{
public:
          void f1 ()               { cout << "f1_D" << endl; }
  virtual void f1 ( const char x ) { cout << "f1_D_char" << endl; }
          void f2 ()               { cout << "f2_D" << endl; }
  virtual void f3 ( const int a )  { cout << "f3_D" << endl; }
};

void g ()
{
  A x;
  D y;
  A *p = &x;

  p->f1 ();
  p->f1 ( '1' );
  p->f1 ( 1 );
  p->f2 ();
  p->f3 ();

  p = &y;
  p->f1 ();
  p->f1 ( '1' );
  p->f1 ( 1 );
  p->f2 ();
  p->f3 ();
}
```

Задача №12

Что будет напечатано в результате вызова функции *main*() в следующем фрагменте программы на Си++?

```
class B
{
```

```
public:
          void f () { cout << "B::f\n"; }
  virtual void g () { cout << "B::g\n"; }
          void h () { cout << "B::h\n"; }
};

class D : public B
{
public:
          void f () { cout << "D::f\n"; }
          void g () { cout << "D::g\n"; }
  virtual void h () { cout << "D::h\n"; }
};

void P ( B * pb,   D * pd )
{
  pb->f ();
  pb->g ();
  pb->h ();
  pd->f ();
  pd->g ();
  pd->h ();
}

int main ()
{
  P ( new B, new D );
  P ( new D, new D );
  return 0;
}
```

Задача №13

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются (считаем, что все объявленные функции где-то описаны).

```
class M
{
public:
  int m;
};

class N
{
public:
  int n;
  void g ();
};

struct A: M, virtual N
{
  char g ( char );
};

struct B: M, virtual N
{
  void g ( int );
};

class D: public A, public B
{
public:
  void f ();
};

void D::f ()
{
  char c = 'a';
  A::m = 1;
  n = 1;
  c = g ( c );
  g ( 2 );
  g ();
}
```

Можно ли исправить ошибки (если они есть), изменяя только описание функции D::f()? Если можно, то каким образом?

Задача №14

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются (считаем, что все объявленные функции где-то описаны).

```
class V
{
public:
  int v;
  void x ( int );
};

class A
{
public:
  int a;
};

struct B: A, virtual V
{
  double x;
};

struct C: A, virtual V
{
  char x;
};

class D: public B, public C
{
public:
  void f ();
};

void D::f ()
{
  v = 1;
  a = 1;
  x ( 6 );
  x = 1.2;
  x = 'a';
}
```

Можно ли исправить ошибки (если они есть), изменяя только описание функции D::f()? Если можно, то каким образом?

Задача №15

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Вычеркните ошибочные конструкции (если они есть). Что будет напечатано при вызове функции *main*()?

```
class complex
{
  double re, im;
public:
                    complex ( double r = 0, double i = 0) { re = r; im = i; }
  operator          double ()       { return re; }
           double   get_re ()       { return re; }
           void     print  () const { cout << "re=" << re << "  im=" << im << endl; }
};

double f ( double x, int i )
{
  cout << "f(double x, int i)" << endl;
  return x*i;
}

complex f ( complex z, int i )
{
  cout << "f(complex z, int i)" << endl;
  return complex ( z.get_re()*i );
}
```

```
int main ()
{
  complex z ( 7, 5 ), y;
  double  x = 1.5, t = 2.8;
  y = f ( z, 2 );
  cout << "y = ";
  y.print();
  t = f ( z, t );
  cout << "t = " << t << endl;
  y = f ( x, 4 );
  cout << " y= ";
  y.print();
  y = f ( z, z );
  cout << "y = ";
  y.print();
  cout << "choice is done!" << endl;
  return 0;
}
```

Задача №16

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Вычеркните ошибочные конструкции (если они есть). Что будет напечатано при вызове функции *main*()?

```
class complex
{
  double   re, im;
public:
  explicit complex ( double r, double i = 0 ) { re = r; im = i; }
           complex ()       { re = 0; im = 0; }
  operator double  ()       { return re; }
  double   get_re  ()       { return re; }
  void     print   () const { cout << "re=" << re << "  im=" << im << endl; }
};

double f ( complex z, int i )
{
  cout << "f(complex z, int i)" << endl;
  return z.get_re() * i;
}

int f ( complex x, complex y )
{
  cout << "f(complex x, complex y)" << endl;
  return x.get_re() * y.get_re();
}

complex f ( complex x, double t )
{
  cout << "f(complex x, double t)" << endl;
  return complex ( x.get_re() * t );
}

int main ()
{
  complex a ( 2, 3 ), b ( 4, 7 );
  double  x = 1.5;
  int i = 15;

  x = f ( a, x );
  cout << "x = " << x << endl;
  x = f ( x, a );
  cout << "x = " << x << endl;
  b = f ( complex ( i ), a );
  cout << "b = ";
  b.print ();
  x = f ( b, ' a' );
  cout << "x = " << x << endl;

  cout << "choice is done!" << endl;
  return 0;
}
```

Задача №17

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Ошибочные конструкции вычеркните из текста программы. Что будет напечатано при вызове функции *main*()?

```
class complex
{
  double re, im;
public:
           complex ( double r = 0, double i = 0 )
           {
             re = r;
             im = i;
             cout << "constructor" << endl;
           }
  operator double ()
           {
             cout << "operator double " << endl;
             return re;
           }
  double   get_re ()       { return re; }
  void     print  () const { cout << "re=" << re << "  im=" << im << endl; }
};

template <class T>
T f ( T& x, T& y )
{
  cout << "template f" << endl;
  return x > y ? x : y;
}

double f ( double x, double y )
{
  cout << "ordinary f" << endl;
  return x > y ? -x : -y;
}

int main ()
{
  complex a ( 2, 5 ), b ( 2, 7 ), c;
  double x = 3.5, y = 1.1;
  int i, j = 8, k = 10;

  c = f ( a, b );
  cout << "c = ";
  c.print ();
  x = f ( a, y );
  cout << "x = " << x << endl;
  i = f ( j, k );
  cout << "i = " << i << endl;
  cout << "choice is done!" << endl;
  return 0;
}
```

Задача №18

Есть ли ошибки в приведенном фрагменте программы на C++? Если есть, то объясните, в чем они заключаются. Ошибочные конструкции вычеркните из текста программы. Что будет напечатано при вызове функции *main*()?

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

int main ()
{
  double x = 1.5, y = 2.8, z;
  int i = 5, j = 12, k;
  char *s1 = "abft";
  char *s2 = "abxde", *s3;

  z = max ( x, y );
  cout << "z = "<< z << endl;
  k = max <int>(i, j);
```

```
  cout << "k = "<< k << endl;
  z = max (x, i);
  cout << "z = "<< z << endl;
  z = max <double> ( y, j );
  cout << "z = "<< z << endl;
  s3 = max  (s1, s2);
  cout << "s3 = "<< s3 << endl;
  cout << "choice is done!" << endl;
  return 0;
}
```

Перегрузите шаблонную функцию max так, чтобы сравнение строк осуществлялось лексикографически (т.е. в соответствии с кодировкой символов).

Задача №19

Какие из следующих утверждений являются верными, а какие — ошибочными? Объясните, в чем заключаются эти ошибки.

- имя объекта класса не может совпадать с именем члена этого класса;
- если функция в базовом классе и в его производном имеет один и тот же профиль и тип возвращаемого значения, то она становится виртуальной;
- если в классе все описанные конструкторы имеют параметры, то компилятор автоматически сгенерирует конструктор умолчания;
- если базовый класс является абстрактным, то в производном классе необходимо описать все его чистые виртуальные функции;
- одноместная операция может быть перегружена функцией-членом без параметров;
- *throw* без выражения может появиться только в try-блоке, где есть обработчик вида *catch* (...){ }.

Задача №20

Перечислите все ситуации, когда вызывается деструктор класса. Приведите примеры каждой из этих ситуаций.

Задача №21

Что произойдет, если в процессе «свертки стека» деструктором класса будет возбуждено исключение?

Задача №22

Написать функцию `g()` с параметром, представляющим собой контейнер-вектор указателей на элементы вещественного типа. Считая от начала контейнера, функция должна обнулять значения, на которое указывают указатели с четными номерами, если значения, на которые указывают указатели с нечетными номерами, отрицательны, а затем распечатывать значения, на которые указывают элементы контейнера в обратном порядке. Функция возвращает число измененных значений.

Задача №23

Написать функцию `g()` с тремя параметрами: непустой и неизменяемый контейнер-вектор типа `vector<double>`, непустой контейнер-список типа `list<double>`, целое число – шаг по первому контейнеру. Функция должна сравнивать элементы списка, выбираемыми от его начала с шагом, равным 1, с элементами вектора, выбираемыми от начала с шагом, равным третьему параметру. Если обнаруживается несовпадение очередной выбранной пары, то в список в текущем месте вставляется отсутствующий

элемент. Изменённый список распечатывается в обратном порядке. Функция возвращает количество элементов, вставленных в список.