

1	2	3	4	5	6	7	8	9	10

1. Если есть ошибки в операторах функции `main()`, исправьте их с помощью операции указания области видимости `::`. Вычеркните операторы, которые не удалось исправить с помощью `::`. Для оставшихся операторов с помощью операции `::` обозначьте ту область видимости, к которой относится вызываемый метод.

```

int x = 4;
class A { int x; public: A (int n = 1);
        int f (int a = 0, int b = 0); };
class B: public A { int x; public: B (int n = 2);
        int f (int a); };
class C: public B { int x; public: C (int n = 3);
        int f (int a, int b);
        int g (A * p); };
A * p; B b; C c;
int main () { p = & b;
             x = c.g (& c);
             x = c.f ();
             x = c.f (x);
             x = c.f (x, 1);
             x = p -> f ();
             x = p -> f (x);
             x = p -> f (x, 1); return 1; }

```

2. Если есть ошибки в приведенной программе, то объясните, в чем они заключаются. Ошибочные операторы или ключевые слова вычеркните (допускается не более двух вычеркиваний). Что будет выдано в стандартный поток вывода при работе получившейся программы?

```

class A {public: static void f (int x) { h (x); cout << "A::f," << x << endl; }
        void g () { cout << "A::g" << endl; }
        void h (int x) { g (); cout << "A::h," << x << endl; }
};
class B: virtual public A {
        public: static void f (int x) { h (x); cout << "B::f," << x << endl; }
        void g () { cout << "B::g" << endl; }
        void h (int x) { g (); cout << "B::h," << x << endl; }
};
int main () { B::f (0); B b; A * p = & b;
            p -> f (1);
            p -> g ();
            p -> h (2);
            A::f (3); return 0; }

```

3. Для каждого вызова перегруженной функции с одним параметром укажите шаг алгоритма, на котором будет выбрана наиболее подходящая функция или выявлена неоднозначность. Для выбранной функции укажите ее прототип.

```

int f (int a = 0) { return a; }
int f (double a) { return a; }
int main () { short int s; int i; bool b; enum e {A, B, C}; float f = 1.0f;
            f ();
            f (s);
            f (f);
            f (b);
            f (A);
}

```

4. Для объектов из задания 1 определить, какие конструкторы и деструкторы и в каком порядке будут выполняться при работе следующего фрагмента программы:

```

int main () { C c; A a = c; struct D { B b; D (): b (5) {} } d; }

```

5. В открытой части класса `cl`, предназначенного для работы со сложными объектами, имеются конструктор и функция преобразования. Для облегчения работы с объектами этого класса создаётся

дополнительный класс *ObjPtr*, в закрытую часть которого включается поле указателя на класс *cl*.

```
template <class T> class cl { /* ... */
public: cl (const T s = 0);
operator T ();
};
template <class T> class ObjPtr { /* ... */ cl<T> * c; public: /* ... */
};
```

Для класса "*ObjPtr*" написать реализацию операций сложения "+" таким образом, чтобы правильными оказались выражения (обязательно использовать и методы, и внешние функции):

```
ObjPtr <int> t1, t2;
/* ... */ (t1 + 3) + (5 + t1) + (t1 + t2) /* ... */
```

6. Что такое виртуальный базовый класс и для чего он используется в Си++? Как синтаксически описываются классы, объявляющие и использующие виртуальные базовые классы?

7. Что будет выдано в стандартный поток вывода при работе следующей программы?

```
void f(X & x, int n);
struct X { X () { try { f(*this, -1); cout << "a"; }
catch (X){ cout << "b"; }
catch (int){ cout << "c"; }
}
X (X &) { cout << "d"; }
virtual ~X () { cout << "e"; } };
struct Y: X { Y () { try { f (*this, 0); cout << "f"; }
catch (Y) { cout << "g"; }
catch (int){ cout << "h"; }
cout << "i"; }
Y (Y &){ cout << "j"; }
~Y () { cout << "k"; } };
void f(X & x, int n) { try { if (n < 0) throw -n;
else if (n == 0) throw x;
else throw n; }
catch (int){ cout << "l"; } }
int main() { try { Y a; }
catch (...){ cout << "m"; return 1; }
cout << "n"; return 0;
}
```

8. Если есть ошибки в следующем фрагменте, то в чем они заключаются? Исправьте ошибки, ничего не удаляя, добавив в общей сложности не более 14 символов.

```
class A { public: int * n; int m; };
class B: public A { public: int * p; };
class C: public A { public: int * c; };
class D: public B: public C { public: int * e; };
int main () {
D fA,
* f = new D;
fA.m =0;
return *((* f).e = & fA.m); }
```

9. Какие виды пользовательских преобразований допустимы в программах на языке Си++? Какие ограничения накладываются на использование таких преобразований при разрешении статического полиморфизма функций? Привести пример фрагмента программы с использованием этих операций.

10. Описать функцию *g()* с двумя параметрами: контейнер-список целых элементов и контейнер-вектор указателей на целые. Функция должна, последовательно проходя по элементам контейнеров от начала к концу вектора и от конца к началу списка, менять местами элементы (целые значения) контейнеров, после чего распечатать в прямом порядке целые значения сначала для списка, затем для вектора. Функция возвращает количество переставленных элементов контейнеров.