

Информатика. Семинар №8

Сколько раз выполнится тело цикла?

```
for (int i(0); i < 100; ++i)
{
    std::cout << i << " ";
    i++;
}
```

Сколько раз выполнится тело цикла?

```
for (int i(0); i < 100; ++i)
{
    std::cout << i << " ";
    i++;
}
```

Ответ: 50

Каким будет значение переменной k
после вызова ф-и f?

```
void f(int& a)
{
    a++;
}

int main()
{
    int k = 0;
    f(k);
}
```

Каким будет значение переменной k
после вызова ф-и f?

```
void f(int& a)
{
    a++;
}

int main()
{
    int k = 0;
    f(k);
}
```

Ответ: 1

Чему эквивалентна (не в объявлении) запись `a[5]` ? где `a` - это указатель или массив и `operator[]` не перегружен

☐

`*(a + 5)`

☐

`5[a]`

☐

`*a + 5`

☐

`(a + 3) [2]`

☐

`*a[5]`

☐

`(a + 2) [3]`

Чему эквивалентна (не в объявлении) запись `a[5]` ? где `a` - это указатель или массив и `operator[]` не перегружен



`*(a + 5)`



`5[a]`



`*a + 5`



`(a + 3)[2]`



`*a[5]`



`(a + 2)[3]`



Скомпилируется ли следующий код?

```
class X {};  
class Y : X {};  
  
int main()  
{  
    X* x = new Y;  
}
```


Скомпилируется ли следующий код?


```
class X {};  
class Y : X {};  
  
int main()  
{  
    X* x = new Y;  
}
```

Нет: наследование private по умолчанию для class'ов, поэтому для конструктора Y конструктор X оказывается private ⇔ недоступен

Что выведет программа?

```
struct A { A() { cout << "A"; } };  
struct B { B() { cout << "B"; } };  
struct C { C() { cout << "C"; } };
```


```
A a;
```

```
int main() {  
    B b;  
}  
C c;
```

Что выведет программа?

```
struct A { A() { cout << "A"; } };  
struct B { B() { cout << "B"; } };  
struct C { C() { cout << "C"; } };
```

```
A a;
```

```
 int main() {
```

```
    B b;
```

```
}
```

```
C c;
```

Ответ: ACB (глобальные
переменные до main)

Что выведет программа?

```
struct A {  
    std::string s;  
    auto p;  
  
    A() { cout << "A"; }  
    A(const std::string& s_, auto p_) :  
        p(p_), s(s_) {}  
    ~A() { cout << "~A"; }  
};  
  
int main() {  
    A("file", "c:/tmp/file.txt");  
}
```

- A
- ~A
- A~A
- Ничего
- Не скомпилируется

Что выведет программа?

```
struct A {  
    std::string s;  
    auto p;  
  
    A() { cout << "A"; }  
    A(const std::string& s_, auto p_) :  
        p(p_), s(s_) {}  
    ~A() { cout << "~A"; }  
};  
  
int main() {  
    A("file", "c:/tmp/file.txt");  
}
```

- A
- ~A
- A~A
- Ничего
- **Не скомпилируется**

1) Не смогли
вывести тип для
auto

2) Список
инициализации:
можно было бы
назвать p(p), s(s)

Что выведет программа?

```
struct A {  
    virtual void f(float x) { cout << x; }  
};  
  
struct B : A {  
    void f(double x) { cout << x + 1; }  
};  
  
int main() {  
    A* x = new B;  
    x->f(1);  
}
```

Что выведет программа?

```
struct A {  
    virtual void f(float x) { cout << x; }  
};
```

```
struct B : A {  
    void f(float x) override { cout << x + 1; }  
};
```

```
int main() {  
    A* x = new B;  
    x->f(1);  
}
```

Старая версия: 1, новая – 2.
override

Что выведет программа?

```
struct A {  
    A() { f(); }  
    virtual void f() { cout << "A"; }  
};
```

```
struct B : A {  
    B() { f(); }  
    void f() { cout << "B"; }  
};
```

```
int main() {  
    A* x = new B;  
    x->f();  
}
```

- ABB
- BBB
- BBA
- AAA
- Не скомпилируется
- «Упадет» при исполнении

Что выведет программа?

```
struct A {  
    A() { f(); }  
    virtual void f() { cout << "A"; }  
};
```

```
struct B : A {  
    B() { f(); }  
    void f() { cout << "B"; }  
};
```

```
int main() {  
    A* x = new B;  
    x->f();  
}
```

- ABB
- BBB
- BBA
- AAA
- Не скомпилируется
- «Упадет» при исполнении

Полиморфизм не работает в конструкторах/деструкторах

Что выведет программа?

```
struct A {  
    int f() { return 1; }  
    int g() { return f() + 1; }  
};
```

```
struct B : A {  
    int f() { return 3; }  
};
```

```
int main() {  
    A a;  
    B b;  
    cout << a.g() << b.g();  
}
```

- 24
- 22
- «упадет»
- не скомпилируется

p.s. Считаем, что выше
написан `#include` и
`using namespace`
нужные.

Что выведет программа?

```
struct A {  
    int f() { return 1; }  
    int g() { return f() + 1; }  
};
```

```
struct B : A {  
    int f() { return 3; }  
};
```

```
int main() {  
    A a;  
    B b;  
    cout << a.g() << b.g();  
}
```

- 24
- 22
- «упадет»
- не скомпилируется

p.s. Считаем, что выше
написан `#include` и
`using namespace`
нужные.

Как обратиться к private'ным переменным класса?

```
class A {  
    int x = 0;  
    friend struct B;  
    friend void g(A& a);  
};
```

```
struct B {  
    void f(A& a) { a.x++; }  
};
```

```
void g(A& a) { a.x++; }
```

Лямбда-функции

```
struct Comparator  
{  
    // TODO  
};
```

```
int main() {  
    std::vector<int> v;  
    // ...  
    std::sort(v.begin(), v.end(), Comparator());  
}
```

Лямбда-функции

```
int count = 0;
```

```
auto comparator = [&](auto a, auto b) {  
    count++;  
    return a > b;  
};
```

```
std::sort(v.begin(), v.end(), comparator);
```

1. Компилятор вместо auto сам «подставит» имя нужного типа.
2. Здесь comparator – функтор (класс, у которого определен оператор «круглые скобки» ())
3. [&] – значит, что ко всем переменным, переданным в качестве аргументов и используемым, обращение будет вестись по ссылке; [=] – по значению; [&a, =b] – первый аргумент по ссылке, второй – по значению.

Лямбда-функции

```
int count = 0;
std::sort(v.begin(), v.end(), [&](auto a, auto b) {
    count++;
    return a > b;
});
```

Где ещё можно использовать лямбда-функции?

```
• std::for_each(v.begin(), v.end(), [&](int x) {  
•   std::cout << x << ' ' ;  
• });
```

```
int count = std::count_if(v.begin(), v.end(), [](int x)  
• {  
•   return x < 5;  
• });
```


Где ещё можно использовать лямбда-функции?

```
1 // copy_if example
2 #include <iostream>      // std::cout
3 #include <algorithm>     // std::copy_if, std::distance
4 #include <vector>        // std::vector
5
6 int main () {
7     std::vector<int> foo = {25,15,5,-5,-15};
8     std::vector<int> bar (foo.size());
9
10    // copy only positive numbers:
11    auto it = std::copy_if (foo.begin(), foo.end(), bar.begin(), [](int i){return !(i<0);} );
12    bar.resize(std::distance(bar.begin(),it)); // shrink container to new size
13
14    std::cout << "bar contains:";
15    for (int& x: bar) std::cout << ' ' << x;
16    std::cout << '\n';
17
18    return 0;
19 }
```

<http://www.cplusplus.com/reference/algorithm/> - если можно воспользоваться какой-то готовой функцией из стандартной библиотеки, то лучше так и сделать, а не писать свою (ещё одну) реализацию.

Упражнение 1

Дано число N и далее N строк. Найти количество уникальных *эквивалентных* строк и вывести их.

P.S. Две строки назовём *эквивалентными*, если после удаления пробелов и перевода символов в нижний регистр они совпадают.

P.P.S. Размер программы должен быть как можно меньше - активно используйте лямбда-функции. Обратите внимание на ϕ -и

<http://www.cplusplus.com/reference/algorithm/transform/>,

`std::sort`,

<http://www.cplusplus.com/reference/algorithm/unique/>

Для работы со строками используйте класс `std::string`.

ПОТОКИ В C++11

```
#include <iostream>
#include <thread>

void func0() {
    std::cout << "Hi! I'm function 0\n";
}

void func1(int x) {
    std::cout << "Hi! I'm function 1." << x << "\n";
}

int main() {
    std::thread t0(func0);
    std::thread t1(func1, 42);

    t0.join();
    t1.join();
}
```

Потоки + лямбда-функции

```
#include <iostream>
#include <thread>

int main() {
    std::thread t0([]() {
        std::cout << "Hi! I'm function 0\n";
    });

    std::thread t1([](int x) {
        std::cout << "Hi! I'm function 1." << x << "\n";
    }, 42);

    t0.join();
    t1.join();
}
```

ПОТОКИ В C++11

```
#include <future>
#include <numeric> // std::accumulate
#include <functional> // std::plus

const int N = 8; const int Size = 1000000;

int main() {
    std::vector<int> v(N * Size);
    for (int& x : v) x = rand() % 10;

    std::chrono::high_resolution_clock::time_point start = std::chrono::high_resolution_clock::now();

    std::vector<std::future<int>> f;
    for (int i = 0; i < N; ++i) {
        f.push_back(std::async([&v, i]() {
            int sum = std::accumulate(v.begin() + i * Size, v.begin() + (i + 1) * Size,
            0, std::plus<int>());
            return sum;
        }));
    }

    for (auto& result : f) std::cout << result.get() << " ";
    std::cout << std::endl;

    std::chrono::duration<double> diff = std::chrono::high_resolution_clock::now() - start;
    std::cout << diff.count() << "s";
}
```

Аналог progress bar`a

```
• std::chrono::milliseconds::span(10);  
• while (f.back().wait_for(span) == std::future_status::timeout)  
• • std::cout << "." << std::flush;
```

Упражнение 2

Методом Монте-Карло посчитать объём сферы в несколько потоков + измерить ускорение.

P.S. Можно и n -мерной:

<https://en.wikipedia.org/wiki/N-sphere>