

Информатика. Семинар №10

Шаблоны

Пример 1. Что выведет программа?

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

int i;

class A
{
public:
    ~A()
    {
        i=10;
    }
};

int foo()
{
    i=3;
    A ob;
    return i;
}

int main()
{
    cout << "i = " << foo() << endl;
    return 0;
}
```

3 или 10?

- $i = 3$
- Как поправить программу, чтобы получилось 10?

i = 10

- `int& foo();`
- `{ A ob; }`

Пример 2

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

int &fun()
{
    static int x = 10;
    return x;
}

int main()
{
    fun() = 30;
    cout << fun();
    return 0;
}
```

- Compiler error: function can't be used as lvalue
- 10
- 30

Пример 2

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

int &fun()
{
    static int x = 10;
    return x;
}

int main()
{
    fun() = 30;
    cout << fun();
    return 0;
}
```

- Compiler error: function can't be used as lvalue
- 10
- 30

Пример 3

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

int main()
{
    int x = 10;
    int& ref = x;
    ref = 20;
    cout << "x = " << x << endl ;
    x = 30;
    cout << "ref = " << ref << endl;
    return 0;
}
```

- x=20 ref=30
- x=20 ref=20
- x=10 ref=30
- x=30 ref=30

Пример 3

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

int main()
{
    int x = 10;
    int& ref = x;
    ref = 20;
    cout << "x = " << x << endl ;
    x = 30;
    cout << "ref = " << ref << endl;
    return 0;
}
```

- x=20 ref=30
- x=20 ref=20
- x=10 ref=30
- x=30 ref=30

Пример 4

```
class Test {  
    int x;  
};  
int main()  
{  
    Test t;  
    cout << t.x;  
    return 0;  
}
```

- 0
- «мусор»
- ошибка компиляции

Пример 4

```
class Test {  
    int x;  
};  
int main()  
{  
    Test t;  
    cout << t.x;  
    return 0;  
}
```

- 0
- «мусор»
- ошибка компиляции
(обращение к private
полю)

Пример 5

Assume that an integer and a pointer each takes 4 bytes. Also, assume that there is no alignment in objects. Predict the output following program.

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

class Test
{
    static int x;
    int *ptr;
    int y;
};

int main()
{
    Test t;
    cout << sizeof(t) << " ";
    cout << sizeof(Test *);
}
```

- 12 4
- 12 12
- 8 4
- 8 8

Пример 5

Assume that an integer and a pointer each takes 4 bytes. Also, assume that there is no alignment in objects. Predict the output following program.

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

class Test
{
    static int x;
    int *ptr;
    int y;
};

int main()
{
    Test t;
    cout << sizeof(t) << " ";
    cout << sizeof(Test *);
}
```

- 12 4
- 12 12
- 8 4
- 8 8

Пример 6

```
#include<iostream>
using namespace std;
class Point {
public:
    Point() { cout << "Constructor called"; }
};

int main()
{
    Point t1, *t2;
    return 0;
}
```

- Ошибка компиляции
- Constructor called (2 раза)
- Constructor called (1 раз)

Пример 6

```
#include<iostream>
using namespace std;
class Point {
public:
    Point() { cout << "Constructor called"; }
};

int main()
{
    Point t1, *t2;
    return 0;
}
```

- Ошибка компиляции
- Constructor called (2 раза)
- Constructor called (1 раз)

Пример 7

```
#include <iostream>
using namespace std;
class A
{
    int id;
    static int count;
public:
    A() {
        count++;
        id = count;
        cout << "constructor for id " << id << endl;
    }
    ~A() {
        cout << "destructor for id " << id << endl;
    }
};

int A::count = 0;

int main() {
    A a[3];
    return 0;
}
```

Пример 7

```
#include <iostream>
using namespace std;
class A
{
    int id;
    static int count;
public:
    A() {
        count++;
        id = count;
        cout << "constructor for id " << id << endl;
    }
    ~A() {
        cout << "destructor for id " << id << endl;
    }
};

int A::count = 0;

int main() {
    A a[3];
    return 0;
}
```

```
constructor for id 1
constructor for id 2
constructor for id 3
destructor for id 3
destructor for id 2
destructor for id 1
```


Пример 8

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

```
int fun(int x = 0, int y = 0, int z)
{   return (x + y + z); }
```

```
int main()
{
    cout << fun(10);
    return 0;
}
```

- 10
- 0
- 20
- Ошибка

КОМПИЛЯЦИИ

Пример 8

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

```
int fun(int x = 0, int y = 0, int z)
{   return (x + y + z); }
```

```
int main()
{
    cout << fun(10);
    return 0;
}
```

- 10
- 0
- 20
- Ошибка компиляции
(параметры по умолчанию
в конце)

Пример 9

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

class Test
{
    static int x;
public:
    Test() { x++; }
    static int getX() {return x;}
};

int Test::x = 0;

int main()
{
    cout << Test::getX() << " ";
    Test t[5];
    cout << Test::getX();
}
```

- 0 0
- 5 5
- 0 5
- Ошибка
КОМПИЛЯЦИИ

Пример 9

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

class Test
{
    static int x;
public:
    Test() { x++; }
    static int getX() {return x;}
};

int Test::x = 0;

int main()
{
    cout << Test::getX() << " ";
    Test t[5];
    cout << Test::getX();
}
```

- 0 0

- 5 5

- 0 5

- Ошибка

КОМПИЛЯЦИИ

Пример 10

```
#include<iostream>

using namespace std;
class Base1 {
public:
    Base1()
    { cout << " Base1's constructor called" << endl; }
};

class Base2 {
public:
    Base2()
    { cout << "Base2's constructor called" << endl; }
};

class Derived: public Base1, public Base2 {
public:
    Derived()
    { cout << "Derived's constructor called" << endl; }
};

int main()
{
    Derived d;
    return 0;
}
```

Пример 10

```
#include<iostream>

using namespace std;
class Base1 {
public:
    Base1()
    { cout << " Base1's constructor called" << endl; }
};

class Base2 {
public:
    Base2()
    { cout << "Base2's constructor called" << endl; }
};

class Derived: public Base1, public Base2 {
public:
    Derived()
    { cout << "Derived's constructor called" << endl; }
};

int main()
{
    Derived d;
    return 0;
}
```

Base1's constructor called
Base2's constructor called
Derived's constructor called

Пример 11

Assume that an integer takes 4 bytes and there is no alignment in following classes, predict the output.

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

```
class base {
    int arr[10];
};
```

```
class b1: public base { };
```

```
class b2: public base { };
```

```
class derived: public b1, public b2 {};
```

```
int main(void)
{
    cout << sizeof(derived);
    return 0;
}
```

- 40
- 80
- 0
- 4

Пример 11

Assume that an integer takes 4 bytes and there is no alignment in following classes, predict the output.

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

```
class base {
    int arr[10];
};
```

```
class b1: public base { };
```

```
class b2: public base { };
```

```
class derived: public b1, public b2 {};
```

```
int main(void)
{
    cout << sizeof(derived);
    return 0;
}
```

- 40

- 80

- 0

- 4

Пример 12

```
#include<iostream>

using namespace std;
class P {
public:
    void print() { cout <<" Inside P"; }
};

class Q : public P {
public:
    void print() { cout <<" Inside Q"; }
};

class R: public Q { };

int main(void)
{
    R r;
    r.print();
    return 0;
}
```

- Inside P
- Inside Q
- Compiler Error:
Ambiguous call to
print()

Пример 12

```
#include<iostream>

using namespace std;
class P {
public:
    void print() { cout <<" Inside P"; }
};

class Q : public P {
public:
    void print() { cout <<" Inside Q"; }
};

class R: public Q { };

int main(void)
{
    R r;
    r.print();
    return 0;
}
```

- Inside P
- Inside Q
(перекрывает
все-все print у
родителей)
- Compiler Error:
Ambiguous call to
print()

Шаблоны функций

- перегрузка функций = название одно и то же, но отличается тип или количество принимаемых параметров

```
int f(int x);  
float f(int x); // error
```

Шаблоны функций

```
int max(int a, int b)
{
    return a > b ? a : b;
}
```

```
float max(float a, float b)
{
    return a > b ? a : b;
}
```

Шаблоны функций

```
template<typename T>
T max(T a, T b)
{
    return a > b ? a : b;
}

int main()
{
    int c = max<int>(1.0f, 2);
    int d = max(1, 2);
    return 0;
}
```

Шаблоны функций

- Шаблонов параметров может быть несколько, могут задаваться типами по умолчанию, могут быть перечислимыми переменными (short, int, long):

```
template<typename T, typename U = T>  
void f(T x, U y);
```


```
template<typename T, int size>  
void f(T x)  
{  
    T a[size];  
    // ...  
}
```

Шаблоны функций

- В качестве целочисленных параметров в шаблоны можно передавать только значения, известные на момент компиляции, т.е. написать так нельзя

```
template<typename T, int size>  
void f(T x);
```

```
int main()  
{  
    int n;  
    f<int, n>(1);  
    return 0;  
}
```



Специализация шаблона

- А что если нашёлся какой-то тип, для которого не определен оператор <?

```
template<>
const MyInt& max(const MyInt& a, const MyInt& b)
{
    return a.Compare(b) ? a : b;
}
```

- Частичную специализацию шаблонов ф-й делать нельзя: нужно явно указывать сразу все шаблонные параметры

Специализация шаблона

```
// A generic sort function
template <class T>
void sort(T arr[], int size)
{
    // code to implement Quick Sort
}

// Template Specialization: A function specialized for char data
template <>
void sort<char>(char arr[], int size)
{
    // code to implement counting sort
}
```

Шаблоны классов. Упражнение 1

Оценить асимптотику
алгоритма сортировки

```
#include <algorithm>
#include <iostream>

template <typename T>
struct Comparator {
    static size_t count;
    bool operator()(const T& x, const T& y) {
        ++count;
        return x < y;
    }
};

template <typename T>
size_t Comparator<T>::count = 0;

int main() {
    std::vector<int> v;
    // ... initialization
    std::sort(v.begin(), v.end(), Comparator<int>());
    return 0;
}
```

Шаблоны классов

```
template<int n>
struct F
{
    enum {
        result = n * F<n - 1>::result
    };
};

template<>
struct F<0>
{
    static const int result = 1;
};

int main() {
    std::cout << F<3>::result << " " << F<4>::result;
    return 0;
}
```

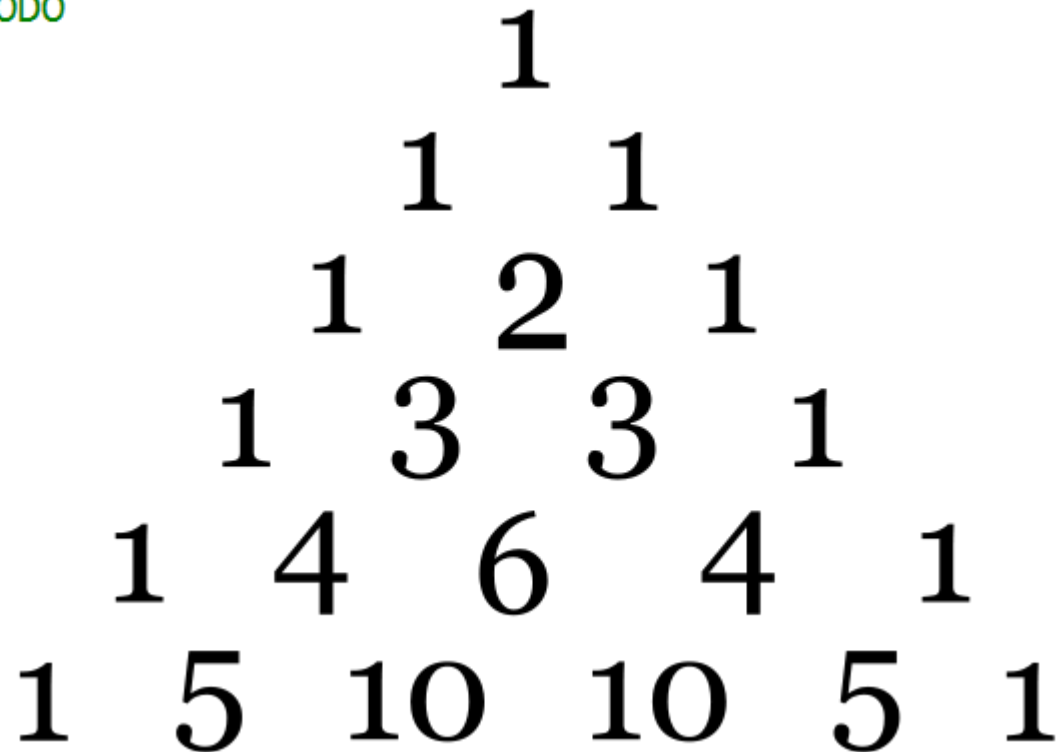
Только enum и static переменные могут быть вычислены на этапе компиляции.
P.S. В C++14 появились constexpr

Шаблоны классов. Упражнение 2

- На этапе компиляции вычислить $C(n, k)$:

```
template<int n, int k>
struct C
{
    ..// TODO
};
```

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$



Шаблоны классов

```
namespace std·  
{  
    ··template·<typename·T,·typename·Allocator>  
    ··class·vector  
    ··{  
        ····//·...  
    ··};  
}
```

```
std::vector<int>·a;  
std::vector<bool>·b;·//·partial·specialization
```

Частичная специализация шаблонов

```
template<typename T>
struct IsPointer
{
    static const bool result = false;
};

template<typename T>
struct IsPointer<T*>
{
    static const bool result = true;
};

int main() {
    std::cout << IsPointer<float>::result << " " << IsPointer<char*>::result;
    return 0;
}
```

Упражнения. 1) Напишите шаблонный класс IsTheSame, для проверки того, одинаковые ли классы у нас, или нет IsTheSame<U, V>::result - аналог std::is_same<U, V>() 2) Выведите числа от 1 до 10 на экран без циклов, рекурсии, стандартных алгоритмов из <algorithm>

Упражнение 3, 4

```
template<typename T>
class Stack
{
public:
    T& Top() const;
    void Pop();
    void Push(const T& elem);
private:
    // TODO
};
```

```
template<typename T, int size>
class FixedSizeStack
{
    // TODO
};
```