National Research University «Higher School of Economics»

Faculty of Computer Science

C++ Questions and Answers

Authors: Khalkechev Roman, Lunev Kirill

Contacts: khalkechev@gmail.com, kirilllunev@gmail.com

Question 1: is this code valid?

```
void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    long a = 10L;
    long b = 100L;
    swap(a, b);
    return 0;
}
```

Example 1

```
void swap(int& a, int& b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int a = 10;
    int b = 100;
    swap(a, b + 3);
    return 0;
}
```

Example 2

Question 1: is this code valid?

```
int cube(int& a) {
    return a * a * a;
}

int main() {
    long a = 3L;
    int b = cube(a);
    int c = cube(10);
    return 0;
}
```

Example 3

```
int cube(const int& a) {
    return a * a * a;
}

int main() {
    long a = 3L;
    int b = cube(a);
    int c = cube(10);
    return 0;
}
```

Example 4

Answer 1:

Example 1: code is not valid

Example 2: code is not valid

Example 3: code is not valid

Example 4: code is valid

Question 2: what will be printed?

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    int a = 0;
    cout << ++++a << endl;
    return 0;
}
```

Example 1

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    int a = 0;
    cout << a++++ << endl;
    return 0;
}
```

Example 2

Question 2: what will be printed?

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    int a = 0;
    cout << +++a << endl;
    return 0;
}
```

Example 3

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    int a = 0;
    cout << ++a++ << endl;
    return 0;
}
```

Example 4

Answer 2:

Example 1: 2

Example 2: code is not valid

Example 3: code is not valid

Example 4: code is not valid

Question 3: what will be printed?

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    int a = 10;
    int b, c;
    b = c = a;
    cout << b << endl;
    cout << c << endl;
    return 0;
}
```

Example 1

```
#include <iostream>
using std::cout;
using std::endl;

int main() {
    int a = 10;
    int b = 0, c = 0;
    b -= c -= a;
    cout << b << endl;
    cout << c << endl;
    return 0;
}
```

Example 2

Answer 3:

Example 1:

10

10

Example 2:

10

-10

Question 4: which function will be called and why?

```
#include <iostream>
void print(double length, double width) {
    std::cout << length << 'x' << width << std::endl;
}
void print(int year, int month) {
    std::cout << year << '.' << month << std::endl;
}
int main() {
    print(3, 5.0);
    return 0;
}
```

Example 1

Question 4: which function will be called and why?

```
double cube(double x) {
    return x * x * x;
}
double cube(double& x) {
    return x = x * x * x;
}
int main() {
    double x = 10.0;
    cube(x);
    return 0;
}
```

Example 2

Question 4: which function will be called and why?

```
#include <iostream>
void foo(char ch) {
    std::cout << "foo(char): " << ch << std::endl;
}
void foo(const char ch) {
    std::cout << "foo(const char): " << ch << std::endl;
}
int main() {
    char ch = 'c';
    foo(ch);
    return 0;
}
```

Example 3

Question 4: which function will be called and why?

```
#include <iostream>
void foo(char* str) {
    std::cout << "foo(char*): " << str << std::endl;
}
void foo(const char* str) {
    std::cout << "foo(const char*): " << str << std::endl;
}
int main() {
    char* s = "C++ is awesome!";
    foo(s);
    return 0;
}
```

Example 4

Question 4: which function will be called and why?

```
#include <cmath>
int foo(double a) {
    return std::abs(a);
}
double foo(double a) {
    return std::fabs(a);
}
int main() {
    double a = -42.0;
    double b = foo(a);
    return 0;
}
```

Example 5

Answer 4:

Example 1: code is not valid, call to 'print' is ambiguous.

Example 2: code is not valid, call to 'cube' is ambiguous.

Example 3: code is not valid, error with multiple definitions of function foo().

Example 4: `void foo(char* str)`, because variable s is not const.

Example 5: code is not valid, because functions that differ only in their return type cannot be overloaded.

Question 5: which function will be called and why?

```cpp
#include <iostream>
template <typename Type> void foo(Type t) {
    std::cout << "foo(Type t)" << std::endl;
}
template <typename Type> void foo(Type* t) {
    std::cout << "foo(Type* t)" << std::endl;
}
int main() {
    int a = 10;
    foo(&a);
    return 0;
}
```

Example 1

Answer 5:

Example 1: `void foo(Type* t)`

Question 6: what will be happen?

```
namespace A {
    struct X { int member = 10; };
    void foo(X& x) { x.member *= 2; }
}
namespace B {
    void foo(A::X& x) { foo(x); }
}
int main() {
    A::X x;
    B::foo(x);
    return 0;
}
```

Example 1

Question 6: what will be happen?

```
namespace A {
    struct X { int member = 10; };
    void foo(X& x) { x.member *= 2; }
}
struct B {
    void foo(A::X& x) { foo(x); }
};
int main() {
    A::X x;
    B b;
    b.foo(x);
    return 0;
}
```

Example 2

Answer 6:

Example 1: code is not valid, call to 'foo' is ambiguous.

Example 2: infinite recursion in function B::foo().

Question 7: is this code valid?

```
struct B {
    int foo(int);
};
struct D : public B {
private:
    int foo(int, double);
};

int main() {
    D d;
    d.foo(10);
    return 0;
}
```

Example 1

Answer 7:

Example 1: code is not valid, because we try to call private function foo of class D, which expects two arguments

Question 8: is this code valid?

```
struct A {
    int size = 32;
    int array[size];
};

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

Example 1

Question 8: is this code valid?

```
struct A {
    const int size = 32;
    int array[size];
};

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

Example 2

Question 8: is this code valid?

```
struct A {
    static int size = 32;
    int array[size];
};

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

Example 3

Question 8: is this code valid?

```
struct A {
    static const int size = 32;
    int array[size];
};

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

Example 4

Question 8: is this code valid?

```
struct A {
    enum { size = 32 };
    int array[size];
};

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

Example 5

Answer 8:

Example 1: code is not valid

Example 2: code is not valid

Example 3: code is not valid

Example 4: code is valid

Example 5: code is valid

Question 9: is this code valid?

```cpp
class TComplex {
     double X, Y;
public:
     TComplex() {}
     TComplex(double x) : X(x), Y(0.0) {}
     TComplex(double x, double y) : X(x), Y(y) {}
     TComplex operator+(const TComplex& complex) {
          complex.X += X;
          complex.Y += Y;
          return complex;
     }
};
int main() {
     TComplex complex;
     TComplex sum = complex + 4.5;
     return 0;
}
```

Example 1

Question 9: is this code valid?

```
class TComplex {
     double X, Y;
public:
     TComplex() {}
     TComplex(double x) : X(x), Y(0.0) {}
     TComplex(double x, double y) : X(x), Y(y) {}
     TComplex operator+(const TComplex& complex) {
          complex.X += X;
          complex.Y += Y;
          return complex;
     }
};
int main() {
     TComplex complex;
     TComplex sum = 4.5 + complex;
     return 0;
}
```

Example 2

Question 9: is this code valid?

```
class TComplex {
    double X, Y;
public:
    TComplex() {}
    TComplex(double x) : X(x), Y(0.0) {}
    TComplex(double x, double y) : X(x), Y(y) {}
    friend TComplex operator+(const TComplex& lhs, const TComplex& rhs);
};
TComplex operator+(const TComplex& lhs, const TComplex& rhs) {
    return TComplex(lhs.X + rhs.X, lhs.Y + rhs.Y);
}

int main() {
    TComplex complex;
    TComplex sum = complex + 4.5;
    return 0;
}
```

Example 3

Question 9: is this code valid?

```
class TComplex {
     double X, Y;
public:
     TComplex() {}
     TComplex(double x) : X(x), Y(0.0) {}
     TComplex(double x, double y) : X(x), Y(y) {}
     friend TComplex operator+(const TComplex& lhs, const TComplex& rhs);
};
TComplex operator+(const TComplex& lhs, const TComplex& rhs) {
     return TComplex(lhs.X + rhs.X, lhs.Y + rhs.Y);
}

int main() {
     TComplex complex;
     TComplex sum = 4.5 + complex;
     return 0;
}
```

Example 4

Answer 9:

Example 1: code is valid

Example 2: code is not valid

Example 3: code is valid

Example 4: code is valid

Question 10: is this code valid?

```
int cube(int& x) {
    x = x * x * x;
    return x;
}
int cube(const int& x) {
    return x * x * x;
}
int main() {
    int a = 2;
    const int b = 3;
    cube(a);
    cube(b);
}
```

Example 1

Question 10: is this code valid?

```
int cube(int x) {
    x = x * x * x;
    return x;
}
int cube(const int x) {
    return x * x * x;
}
int main() {
    int a = 2;
    const int b = 3;
    cube(a);
    cube(b);
}
```

Example 2

Answer 10:

Example 1: code is valid

Example 2: code is not valid

Question 11: what will be printed?

```
#include <iostream>
struct Y;
struct X {
	double Data;
	X(double data) : Data(data) {}
	X(const Y& y) { std::cout << "X(const Y& y)\n"; }
	X(const X& x) { std::cout << "X(const X& x)\n"; }
	X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
	int Data;
};
int main() {
	Y y = {10};
	X x = y;
}
```

Example 1

Question 11: what will be printed?

```
#include <iostream>
struct Y;
struct X {
	double Data;
	X(double data) : Data(data) {}
	X(const Y& y) { std::cout << "X(const Y& y)\n"; }
	X(const X& x) { std::cout << "X(const X& x)\n"; }
	X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
	int Data;
	operator X() { std::cout << "operator X()\n"; return X(Data); }
};
int main() {
	Y y = {10};
	X x = y;
}
```

Example 2

Question 11: what will be printed?

```
#include <iostream>
struct Y;
struct X {
     double Data;
     X(double data) : Data(data) {}
     X(const Y& y) { std::cout << "X(const Y& y)\n"; }
     X(const X& x) { std::cout << "X(const X& x)\n"; }
     X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
     int Data;
     operator X() { std::cout << "operator X()\n"; return X(Data); }
};
int main() {
     const Y y = {10};
     X x = y;
}
```

Example 3

Question 11: what will be printed?

```
#include <iostream>
struct Y;
struct X {
     double Data;
     X(double data) : Data(data) {}
     X(const Y& y) { std::cout << "X(const Y& y)\n"; }
     X(const X& x) { std::cout << "X(const X& x)\n"; }
     X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
     int Data;
     operator X() const { std::cout << "operator X()\n"; return X(Data); }
};
int main() {
     Y y = {10};
     X x = y;
}
```

Example 4

Question 11: what will be printed?

```
#include <iostream>
struct Y;
struct X {
	double Data;
	X(double data) : Data(data) {}
	X(const Y& y) { std::cout << "X(const Y& y)\n"; }
	X(const X& x) { std::cout << "X(const X& x)\n"; }
	X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
	int Data;
	operator X() const { std::cout << "operator X()\n"; return X(Data); }
};
int main() {
	const Y y = {10};
	X x = y;
}
```

Example 5

Question 11: what will be printed?

```cpp
#include <iostream>
struct Y;
struct X {
    double Data;
    X(double data) : Data(data) {}
    X(Y& y) { std::cout << "X(const Y& y)\n"; }
    X(const X& x) { std::cout << "X(const X& x)\n"; }
    X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
    int Data;
    operator X() const { std::cout << "operator X()\n"; return X(Data); }
};
int main() {
    Y y = {10};
    X x = y;
}
```

Example 6

Question 11: what will be printed?

```
#include <iostream>
struct Y;
struct X {
	double Data;
	X(double data) : Data(data) {}
	explicit X(const Y& y) { std::cout << "X(const Y& y)\n"; }
	X(const X& x) { std::cout << "X(const X& x)\n"; }
	X& operator=(const X& x) { std::cout << "operator=\n"; Data = x.Data; return *this; }
};
struct Y {
	int Data;
	operator X() const { std::cout << "operator X()\n"; return X(Data); }
};
int main() {
	Y y = {10};
	X x = y;
}
```

Example 7

Answer 11:

Example 1: X(const Y& y)

Example 2: operator X()

Example 3: X(const Y& y)

Example 4: code is not valid

Example 5: code is not valid

Example 6: X(const Y& y)

Example 7: operator X()

Question 12: is this code valid?

```
class A {
public:
    A* Clone();
};
A* A::Clone() {
    return this;
}
int main() {
    A a;
    A* pa = a.Clone();
    return 0;
}
```

Example 1

Question 12: is this code valid?

```
class A {
public:
    A* Clone() const;
};
A* A::Clone() const {
    return this;
}
int main() {
    A a;
    A* pa = a.Clone();
    return 0;
}
```

Example 2

Question 12: is this code valid?

```cpp
class A {
public:
    const A* Clone() const;
};
const A* A::Clone() const {
    return this;
}
int main() {
    A a;
    const A* pa = a.Clone();
    return 0;
}
```

Example 3

Question 12: is this code valid?

```
class A {
public:
    const A* Clone();
};
const A* A::Clone() {
    return this;
}
int main() {
    A a;
    const A* pa = a.Clone();
    return 0;
}
```

Example 4

Answer 12:

Example 1: code is valid

Example 2: code is not valid

Example 3: code is valid

Example 4: code is valid

Question 13: what will be printed?

```cpp
#include <iostream>
struct Base {
    virtual void foo(int x = 10) { std::cout << "Base: " << x << std::endl; }
};
struct Derived : public Base {
    void foo(int x = 20) override { std::cout << "Derived: " << x << std::endl; }
};
int main() {
    Base base;
    Derived derived;
    Base* pBase = &derived;
    base.foo();
    derived.foo();
    pBase->foo();
    return 0;
}
```

Example 1

Answer 13:

Example 1:

```
Base: 10

Derived: 20

Derived: 10
```

Question 14: what will be printed?

```cpp
#include <iostream>
namespace N {
    void f (double) { std::cout << "void f(double)\n"; }
}
using N::f;
namespace N {
    void f (int) { std::cout << "void f(int)\n"; }
}

int main() {
    int x = 10;
    f(x);
    return 0;
}
```

Example 1

Answer 14:

Example 1: void f(double)

Question 15: is this code valid?

```
#include <iostream>
namespace N {
    class X {};
}
using namespace N;
void f() {
    X x;
    Y y;
}
namespace N {
    class Y {};
}
int main() {
    return 0;
}
```

Example 2

Answer 15:

Example 1: code is not valid

Question 16: is this code valid?

```
#include <iostream>
class TBase {
public:
    virtual ~TBase() = 0;
};
class TDerived : public TBase {
public:
    virtual ~TDerived();
};
TDerived::~TDerived() {}
int main() {
    TDerived derived;
    return 0;
}
```

Example 1

Question 16: is this code valid?

```
#include <iostream>
class TBase {
public:
    virtual ~TBase() = 0;
};
TBase::~TBase() {}
class TDerived : public TBase {
public:
    virtual ~TDerived();
};
TDerived::~TDerived() {}
int main() {
    TDerived derived;
    return 0;                         Example 2
}
```

Answer 16:

Example 1: code is not valid

Example 2: code is valid

Question 17: is this code valid?

```
#include <new>
template<typename T1, typename T2>
void Construct(T1* p, const T2& value) {
    new(p) T1(value);
}
void f(double* p) {
    Construct(p, 5);
}

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

Example 1

Question 17: is this code valid?

```
#include <new>
template<typename T1>
void Construct(T1* p, const T1& value) {
    new(p) T1(value);
}
void f(double* p) {
    Construct(p, 5);
}

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

Example 2

Question 17: is this code valid?

```
#include <new>
template<typename T1>
void Construct(T1* p, const T1& value) {
    new(p) T1(value);
}
void f(double* p) {
    Construct<double>(p, 5);
}

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

Example 3

Answer 17:

Example 1: code is valid

Example 2: code is not valid

Example 3: code is valid

Question 18: what will be printed?

```
#include <iostream>
template<typename T>
void foo(T) { std::cout << "foo<T>(T)\n"; }

template<typename T>
void foo(T*) { std::cout << "foo<T>(T*)\n"; }

template<>
void foo<int>(int*) { std::cout << "foo<int>(int*)\n"; }

int main() {
    int* p;
    foo(p);
    return 0;
}
```

Example 1

Question 18: what will be printed?

```
#include <iostream>
template<typename T>
void foo(T) { std::cout << "foo<T>(T)\n"; }

template<typename T>
void foo(T*) { std::cout << "foo<T>(T*)\n"; }

template<>
void foo<int*>(int*) { std::cout << "foo<int*>(int*)\n"; }

int main() {
    int* p;
    foo(p);
    return 0;
}
```

Example 2

Answer 18:

Example 1: foo<int>(int*)

Example 2: foo<T>(T*)

Question 19: is this code correct?

```
#include <iostream>
#include <string>
struct A {
    A(const std::string& s) { std::cout << s << std::endl; }
    std::string foo() { return "Hello, World!"; }
};
struct B : public A {
    std::string s;
public:
    B() : A(foo()) {}
};
int main() {
    B b;
    return 0;
}
```

Example 1

Question 19: is this code correct?

```
#include <iostream>
#include <string>
struct A {
    A(const std::string& s) { std::cout << s << std::endl; }
    std::string foo() { return "Hello, World!"; }
};
struct B : public A {
    std::string s;
public:
    B() : A(s) {}
};
int main() {
    B b;
    return 0;
}
```

Example 2

Question 19: is this code correct?

```
#include <iostream>
#include <string>
struct A {
    A(const std::string& s) { std::cout << s << std::endl; }
    std::string foo() { return "Hello, World!"; }
};
struct B : public A {
    std::string s;
public:
    B() : A(s = foo()) {}
};
int main() {
    B b;
    return 0;
}
```

Example 2

Answer 19:

Example 1: code is incorrect

Example 2: code is incorrect

Example 3: code is incorrect

Question 20: in what sequence will be called constructors?

```
class V1 {};
class V2 {};
class B {};
class Member {};

class B1 : virtual public V1 {};
class B2 : public B, virtual public V2 {};

class D : public B1, public B2 {
    Member M;
};
int main() {
    D d;
    return 0;
}
```

Example 1

Answer 20:

Example 1: in this sequence: V1(), V2(), B1(), B(), B2(), Member()

Question 21: what will be printed?

```
#include <iostream>
class Animal {
public:
    void Eat() { std::cout << "Omnomnom" << std::endl; }
};
class Mammal : public Animal {};
class WingedAnimal : public Animal {};
class Bat : public Mammal, public WingedAnimal {};
int main() {
    Bat bat;
    bat.Eat();
    return 0;
}
```

Example 1

Question 21: what will be printed?

```
#include <iostream>
class Animal {
public:
    virtual void Eat() { std::cout << "Omnomnom" << std::endl; }
};
class Mammal : public Animal {};
class WingedAnimal : public Animal {};
class Bat : public Mammal, public WingedAnimal {};
int main() {
    Bat bat;
    bat.Eat();
    return 0;
}
```

Example 2

Question 21: what will be printed?

```
#include <iostream>
class Animal {
public:
    void Eat() { std::cout << "Omnomnom" << std::endl; }
};
class Mammal : public virtual Animal {};
class WingedAnimal : public virtual Animal {};
class Bat : public Mammal, public WingedAnimal {};
int main() {
    Bat bat;
    bat.Eat();
    return 0;
}
```

Example 3

Answer 21:

Example 1: code is not valid

Example 2: code is not valid

Example 3: "Omnomnom"

Question 22: is this code valid?

```
#include <iostream>
class A {
public:
    void foo(double d) { std::cout << d << std::endl; }
private:
    void foo(int i) { std::cout << i << std::endl; }
};

int main() {
    A a;
    a.foo(10);
}
```

Example 1

Answer 22:

Example 1: code is not valid

Question 23: what will be printed?

```
#include <iostream>
class A {
    int privateValue;
public:
    A(int value) : privateValue(value) {}
    template<typename T> void foo(const T&) {}
    void show() { std::cout << privateValue << std::endl; }
};
class Y {};
template<> void A::foo(const Y&) { privateValue = 10; }
int main() {
     A a(0);
     a.show();
     a.foo(Y());
     a.show();
}
```

Example 1

Answer 23:

Example 1:

0

10

Question 24: what will be printed?

```
#include <iostream>
class A {
    int Member;
    void PrivateFunction(int value) { Member = value; }
public:
    typedef void (A::*FuncPointer)(int);
    FuncPointer PublicFunction() const { return &A::PrivateFunction; }
    int GetMember() const { return Member; }
};
int main() {
    A a;
    A::FuncPointer funcPointer = a.PublicFunction();
    (a.*funcPointer)(42);
    std::cout << a.GetMember() << std::endl;
}
```

Example 1

Answer 24:

Example 1: 42

Question 25: what will be printed?

```cpp
#include <iostream>

#define max(a, b) a > b ? a : b

int main() {
    int a = 10;
    int b = 5;
    std::cout << max(a, b) << std::endl;
}
```

Example 1

Question 25: what will be printed?

```
#include <iostream>

#define max(a, b) a > b ? a : b

int main() {
    int a = 10;
    int b = 5;
    int m = max(a, b);
    std::cout << m << std::endl;
}
```

Example 2

Question 25: what will be printed?

```
#include <iostream>

#define max(a, b) a > b ? a : b

int main() {
    int a = 10;
    int b = 5;
    int m = max(a, b) + 2;
    std::cout << m << std::endl;
}
```

Example 3

Question 25: what will be printed?

```
#include <iostream>

#define max(a, b) (a > b ? a : b)

int main() {
    int a = 10;
    int b = 5;
    int m = max(a, b) + 2;
    std::cout << m << std::endl;
}
```

Example 4

Question 25: what will be printed?

```
#include <iostream>

#define max(a, b) (a > b ? a : b)

int main() {
    int a = 10;
    int b = 5;
    int m = max(++a, b);
    std::cout << m << std::endl;
}
```

Example 5

Answer 25:

Example 1: code is not valid

Example 2: 10

Example 3: 10

Example 4: 12

Example 5: 12

The end