Choice Questions

1

Choose the respective delete operator usage for the expression 'ptr=new int[100]'.
a) `delete ptr;`
b) `delete ptr[];`
c) `delete[] ptr;`
d) `[]delete ptr;`

> Answer : C

2

With respective to streams >> (operator) is called as
a) Insertion operator
b) Extraction operator
c) Right shift operator
d) Left shift operator

> Answer : B
> Explanation: It extracts the data from stream into variables.

3

What will be the output of the following C++ code?

```
#include<iostream>
using namespace std;
int main ()
{
   int cin;
   cin >> cin;
   cout << "cin: " << cin;
   return 0;
}
```

a) Segmentation fault

b) Nothing is printed

c) Error

d) cin: garbage value

> Answer: d
>
> Explanation: cin is a variable hence overrides the cin object. cin >> cin has no meaning so no error.

4

What will be the output of the following C++ code?

```
#include <iostream>
using namespace std;
void square (int *x, int *y)
{
    *x = (*x) * --(*y);
}
int main ( )
{
    int number = 30;
    square(&number, &number);
    cout << number;
    return 0;
}
```

a) 30

b) Error

c) Segmentation fault

d) 870

> Answer: d
>
> Explanation: As we are passing value by reference therefore the change in the value is reflected back to the passed variable number hence value of number is changed to 870.

5

Define

```
std::map<std::string, int> kv;
```

What is the static type of x in the scope for loop?

```
for(auto x:kv){}
```

a) `std::pair<const std::string ,const int >`
b) `std::pair<const std::string , int>`
c) `std::pair<std::string, int>`
d) `std::pair<const std::string, int>::iterator`

> Answer: B
> Explanation: The declaration `std::map<std::string, int> kv;` declares a map named `kv` which has keys of type `std::string` and values of type `int`.
> In the scope of the for loop `for(auto x:kv){}`, the static type of `x` is `std::pair<const std::string, int>`. This is because the elements of a `std::map` are key-value pairs, where the key is of type `const std::string` and the value is of type `int`.
> When `auto` is used in a range-based for loop, the type of the loop variable `x` is deduced from the type of the elements in the range expression. In this case, since `kv` is a

> `std::map<std::string, int>`, the elements are of type `std::pair<const std::string, int>`. Therefore, `auto x` is deduced to be of type `std::pair<const std::string, int>`.

6

Which of the following options will result in a different result than the other options?
a) `auto a(char());`
b) `auto a(char(0));`
c) `auto a(char{});`
d) `auto a(char{0});`

> Answer: A

7

Which of the following given can be considered as the correct output of the following C ++ code?

```
#include<iostream>
usingnamespace std;
int main()
{
    int x=5;
    int y=5;
    auto check =[&x]()
    {
        x =10;
        y =10;
    }
    check();
    cout<<"Value of x: "<<x<<endl;
    cout<<"Value of y: "<<y<<endl;
    return0;
}
```

a) It will result in an Error
b) Value of x: 10
c) Value of y: 5
d) It will obtain Segmentation fault

> Answer: a
> Explanation: If you look at the above-given program carefully, you will find that the lambda expression does not capture the value of variable "y" at all. Besides, it tries to access the value of the external variable y. Thus the above-given program will obtain or result in an Error. Therefore the correct answer is A.

8

Which of the following can be considered as the correct syntax of for loop?
a) `for(initialization; condition; increment/decrement operator){}`
b) `for(initialization, condition; increment/decrement operator){}`
c) `for(initialization; increment/decrement operator;condition;{}`
d) None of the above

> Answer: a
> Description: Usually, the for loop contains three statements and the working of these statementsis shown in the following syntax:
>
> for (statement 1; statement 2; statement 3) {
> // code block to be executed
> }
> statement1: initialzation
> statement2: condition to be check
> statement3: increment or decrement operator

9

What is the difference between delete and delete[] in C++?
a) delete is syntactically correct but delete[] is wrong and hence will give an error if used in any case
b) delete is used to delete normal objects whereas delete[] is used to pointer objects
c) delete is a keyword whereas delete[] is an identifier
d) delete is used to delete single object whereas delete[] is used to multiple(array/pointer of) objects

> Answer: d
> Explanation: delete is used to delete a single object initiated using new keyword whereas delete[] isused to delete a group of objects initiated with the new operator

10

(i) 'ios' is the base class of 'istream'

(ii) All the files are classified into only 2 types. (1) Text Files (2) Binary Files.

A - Only (i) is true

B - Only (ii) is true

C - Both (i) & (ii) are true

D - Both (i) & (ii) are false

> Answer : C

11

What is the output of the following program?

```
#include<iostream>

using namespace std;
int main() {
   int const a = 5;
   a++;
   cout << a;
}
```

A - 5

B - 6

C - Runtime error

D - Compile error

> Answer： D
>
> constant variable cannot be modified

12

What is the output of the following program?

```
#include<iostream>

using namespace std;
int x = 5;

int &f() {
    return x;
}
int main() {
   f() = 10;
   cout<<x;
}
```

A - 5

B - Address of 'x'

C - Compile error

D - 10

> Answer： D
>
> A function can return reference, hence it can appear on the left hand side of the assignment operator.

Fill in the blank

1. `std::cin` is an input stream. It has type _____.

   > Answer： `std::istream`

2. There are two types of containers: Associative and _____.

   > Answer： Sequence

3. Fill in the types (`int`、`std::string`、`double`、`void`)

```
_______ a = "test";
_______ b = 3.2 * 5 - 1;
_______ c = 5 / 2;
_______ d(int foo) { return foo / 2; }
_______ e(double foo) { return foo / 2; }
_______ f(double foo) { return int(foo / 2); }
_______ g(double c) {
    std::cout << c << std::endl;
}
```

> Answer： `std::string`、`double`、`int`、`double`、`int`、`void`

4. C++ is a statically typed language or dynamically typed language? _____.

> Answer： statically typed

5. Does vector check bounds in this scenario?(Yes or No) _____.

```
std::vector<int> vec;
...
int k = vec[i];
```

> Answer： No

Short Answer Question

1.Briefly describe the process of compiling 'a.cpp' to generate executable file 'a.out' (with g++)

> 参考答案:
>
> 预处理
> 编译
> 汇编
> 链接

2.Write the command to compile and link the following files into an executable file 'main. out' (using g++)

`main.cpp`

```
// main.cpp
#include "hello.h"
int main() {
    quiz1::printhello(); // 修正
}
```

`hello.h`

```
// hello.h
namespace quiz1 {
    void printhello();
}
```

`hello.cpp`

```
// hello.cpp
#include "hello.h"
#include <iostream>
void quiz1::printhello() {
    std::cout << "Hello Quiz1" << std::endl;
}
```

> 参考答案:
>
> ```
> # 方法1
> g++ main.cpp hello.cpp -o main.out
>
> # 方法2
> g++ -c main.cpp -o main.o
> g++ -c hello.cpp -o hello.o
> g++ main.o hello.o -o main.out
> ```

3.Which of `const`, `enum`, and `define` has an address in the following program, and provide a brief explanation

```
// test_const
#include <iostream>
using namespace std;
struct A {
    #define p "hello"
};

class C {
public:
    static const int NUM = 3;
    enum con {
        NUM1 = 3
    };
};

#define MAX(a,b) ((a) > (b) ? (a) : (b))
template<typename T>
inline int Max(const T& a, const T& b){
    return (a>b ? a:b);
}
const int C::NUM;
int main() {
    cout << p << endl;
    C c;
```

```
    cout << &c.NUM << endl;
    cout << C::NUM1 << endl;
    cout << &C:NUM1 << endl;

    int a=5, b=0;
    cout<<MAX(++a, b)<<endl;
    cout<<MAX(++a, b+10)<<endl;
    a=5,b=0;
    cout<<Max(++a,b)<<endl;
}
```

> 参考答案:
>
> const 有地址,enum与#define没有地址
>
> 1. const 定义的实际是一个变量,const只限定它不能被修改,所有变量都可在程序运行时获取其地址
> 2. enum类型中的枚举项只是enum类型声明的一部分,它不是定义出来的变量,所以不能取地址
> 3. #define出来的是宏,它是预处理的东西,预处理后的编译阶段已经不存在,所以也不可能获取宏的地址

4.Briefly describe the usage and function of `static`

> 参考答案:
>
> 1. 用于全局变量：变量只能在本文件使用
> 2. 用于局部变量：变量只被初始化一次
> 3. 用于函数：函数只能在本文件中使用

5.Using structured binding, implement the following functions using only one line of code within the function

```
#include <iostream>
#include <map>
#include <string>
#include <functional>

template <typename Key, typename Value, typename F>
    void update(std::map<Key, Value>& m, F foo) {
    // TODO:
}
int main() {
    std::map<std::string, long long int> m {
        {"a", 1},
        {"b", 2},
        {"c", 3}
    };
    update(m, [](std::string key) {
        return std::hash<std::string>{}(key);
    });
    for (auto&& [key, value] : m)
        std::cout << key << ":" << value << std::endl;
}
```

Answer:

```cpp
#include <iostream>
#include <map>
#include <string>
#include <functional>

template <typename Key, typename Value, typename F>
void update(std::map<Key, Value>& m, F foo) {
    for (auto&& [key, value] : m ) value = foo(key);
}

int main() {
    std::map<std::string, long long int> m {
        {"a", 1},
        {"b", 2},
        {"c", 3}
    };
    update(m, [](std::string key) -> long long int {
        return std::hash<std::string>{}(key);
    });
    for (auto&& [key, value] : m)
        std::cout << key << ":" << value << std::endl;
}
```