C++ Pointers & Functions Job Interview Questions And Answers

Interview Questions Answers

https://interviewquestionsanswers.org/

About Interview Questions Answers

Interview Questions Answers . ORG is an interview preparation guide of thousands of Job Interview Questions And Answers, Job Interviews are always stressful even for job seekers who have gone on countless interviews. The best way to reduce the stress is to be prepared for your job interview. Take the time to review the standard interview questions you will most likely be asked. These interview questions and answers on C++ Pointers & Functions will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts.

If you find any **question or answer** is incorrect or incomplete then you can **submit your question or answer** directly with out any registration or login at our website. You just need to visit C++ Pointers & Functions Interview Questions And Answers to add your answer click on the *Submit Your Answer* links on the website; with each question to post your answer, if you want to ask any question then you will have a link *Submit Your Question*; that's will add your question in C++ Pointers & Functions category. To ensure quality, each submission is checked by our team, before it becomes live. This C++ Pointers & Functions Interview preparation PDF was generated at **Wednesday 29th November, 2023**

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter for latest Jobs and interview preparation guides.
https://twitter.com/InterviewQA

If you need any further assistance or have queries regarding this document or its material or any of other inquiry, please do not hesitate to contact us.

Best Of Luck.

Interview Questions Answers.ORG Team
https://InterviewQuestionsAnswers.ORG/
Support@InterviewQuestionsAnswers.ORG

C++ Pointers & Functions Interview Questions And Answers Guide.

Question - 1:

Where does the execution of the program starts?
a) user-defined function
b) main function
c) void function
d) none of the mentioned

Ans:

b) main function

View All Answers

Question - 2:

What are mandatory parts in function declaration?
a) return type,function name
b) return type,function name,parameters
c) both a and b
d) none of the mentioned

Ans:

a) return type,function name

View All Answers

Question - 3:

How many max number of arguments can present in function in c99 compiler?
a) 99
b) 90
c) 102
d) 127

Ans:

d) 127

View All Answers

Question - 4:

Which is more effective while calling the functions?
a) call by value
b) call by reference
c) call by pointer
d) none of the mentioned

Ans:

b) call by reference

View All Answers

Question - 5:

The output of this program?

```
#include <iostream>
using namespace std;
int main()
{
    int arr[] = {4, 5, 6, 7};
    int *p = (arr + 1);
```

```
    cout << *p;
    return 0;
  }
```

a) 4
b) 5
c) 6
d) 7

Ans:

b) 5

View All Answers

Question - 6:

Output of this program?

```
  #include <iostream>
  using namespace std;
  int main()
  {
    int i;
    char *arr[] = {"C", "C++", "Java", "VBA"};
    char *(*ptr)[4] = &arr;
    cout << ++(*ptr)[2];
    return 0;
  }
```

a) ava
b) java
c) c++
d) compile time error

Ans:

a) ava

View All Answers

Question - 7:

What is the output of this program?

```
  #include <iostream>
  using namespace std;
  int main()
  {
    int a[2][4] = {3, 6, 9, 12, 15, 18, 21, 24};
    cout << *(a[1] + 2) << *(*(a + 1) + 2) << 2[1[a]];
    return 0;
  }
```

a) 15 18 21
b) 21 21 21
c) 24 24 24
d) Compile time error

Ans:

b) 21 21 21

View All Answers

Question - 8:

What is meaning of following declaration?
int(*p[5])();
a) p is pointer to function.
b) p is array of pointer to function.
c) p is pointer to such function which return type is array.
d) p is pointer to array of function

Ans:

b) p is array of pointer to function.

View All Answers

Question - 9:

What is size of generic pointer in c?
a) 0
b) 1
c) 2
d) Null

Ans:

c) 2

View All Answers

Question - 10:

How many minimum number of functions are need to be presented in c++?
a) 0
b) 1
c) 2
d) 3

Ans:

b) 1

View All Answers

Question - 11:

What is the scope of the variable declared in the user definied function?
a) whole program
b) only inside the {} block
c) both a and b
d) none of the mentioned

Ans:

b) only inside the {} block

View All Answers

Question - 12:

What is the output of this program?

```
#include <iostream>
using namespace std;
void fun(int x, int y)
{
   x = 20;
   y = 10;
}
int main()
{
   int x = 10;
   fun(x, x);
   cout << x;
   return 0;
}
```

a) 10
b) 20
c) compile time error
d) none of the mentioned

Ans:

a) 10

View All Answers

Question - 13:

What is the output of this program?

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

a) hai
b) haihai
c) compile time error
d) none of the mentioned

Ans:

c) compile time error

View All Answers

Question - 14:

Which of the following is used to terminate the function declaration?
a) :
b))
c) ;
d) none of the mentioned

Ans:

c) ;

View All Answers

Question - 15:

Can you please explain the difference between Pointer to constant and pointer constant?

Ans:

Pointer to constant points to a value that does not change and is declared as:
const type * name
type is data type
name is name of the pointer
e.g: const char *p;
pointer to constant can not be used to change the value being pointed to. Therefore:
char ch = â€˜A';
const char *p = &ch;
*p = â€˜B';
is not allowed. The program will throw an error.
Pointer Constant (or constant pointer) is a pointer which you don't want to be pointed to a different value. That is, the location stored in the pointer can not change. We can not change where the pointer points. It is declared as:
type * const name
type is data type
name is name of the pointer
eg: char * const p
Since the location to which a const pointer points to can not be changed, the following code:
char ch1 = â€˜A';
char ch2 = â€˜B';
char * const p = &ch1;
p = &ch2;
will throw an error since address stored in p can not be changed.

View All Answers

Question - 16:

Explain const pointer and const reference?

Ans:

const pointer is a pointer which you don't want to be pointed to a different value. That is, the location stored in the pointer can not change. We can not change where the pointer points. It is declared as:
type * const name
type is data type
name is name of the pointer
eg: char * const p
Since the location to which a const pointer points to can not be changed, the following code:
char ch1 = â€˜A';
char ch2 = â€˜B';
char * const p = &ch1;
p = &ch2;
will throw an error since address stored in p can not be changed.
const reference:
const references allow you to specify that the data referred to won't be changed. A const reference is actually a reference to const. A reference is inherently const, so when we say const reference, it is not a reference that can not be changed, rather it's a reference to const. Once a reference is bound to refer to an object, it can not be bound to refer to another object. For example:
int &ri = i;
binds ri to refer to i. Then assignment such as:
ri = j;
doesn't bind ri to j. It assigns the value in j to the object referenced by ri, ie i;
This means, if we pass arguments to a function by const references; the function can not change the value stored in those references. This allows us to use const references as a simple and immediate way of improving performance for any function that currently takes objects by value without having to worry that your function might modify the data. The compiler will throw an error if the function tries to modify the value of a const reference.

View All Answers

Question - 17:

Explain function pointers?

Ans:

A function has a physical location in the memory which is the entry point of the function. And this is the address used when a function is called. This address can be assigned to a pointer. Once a pointer points to a function, the function can be called through that pointer. Function pointers also allow functions to be passed as arguments to other functions. The address of a function is obtained by using the function's name without any parenthesis or arguments.
Consider following example:

```
#include <iostream>
using namespace std;
void check(char *a, char *b, int (*cmp) (const char*, const*));
int numcmp(const char *a, const char *b);
int main()
{
    char s1[80], s2[80];
    gets (s1);
    gets (s2);
    if (isalpha(*s1))
        check(s1, s2, strcmp);
```

```
    else
        check(s1, s2, numcmp);
    return 0;
}
void check(char *a, char *b, int (*cmp) (const char*, const*))
{
    cout <<â€•Testing for equality
â€•;
    if(!(*cmp)(a,b))
        cout <<â€œEqualâ€•;
    else
        cout <<â€œ Not Equalâ€•;
}
int numcmp(const char *a, const char *b)
{
    if(atoi(a) == atoi(b))
        return 0;
    else
        return 1;
}
```

In this function, if you enter a letter, strcmp() is passed to check() otherwise numcmp() is used.

View All Answers

Question - 18:

Explain pointer with examples?

Ans:

A pointer is a variable that holds a memory address. This address is the location of another object (typically, a variable) in memory. That is, if one variable contains the address of another variable, the first variable is said to point to the second.
A pointer declaration consists of a base type, an *, and the variable name. The general form of declaring a pointer variable is:
type *name;
type is the base type of the pointer and may be any valid type.
name is the name of pointer variable.
The base type of the pointer defines what type of variables the pointer can point to.

View All Answers

Question - 19:

What is Pointer Constant?

Ans:

Pointer Constant (or constant pointer) is a pointer which you don't want to be pointed to a different value. That is, the location stored in the pointer can not change. We can not change where the pointer points. It is declared as:
type * const name
type is data type
name is name of the pointer
eg: char * const p
Since the location to which a const pointer points to can not be changed, the following code:

```
char ch1 = â€˜A';
char ch2 = â€˜B';
char * const p = &ch1;
p = &ch2;
```

will throw an error since address stored in p can not be changed.

View All Answers

Question - 20:

What is Pointer to constant?

Ans:

Pointer to constant points to a value that does not change and is declared as:
const type * name
type is data type
name is name of the pointer
e.g: const char *p;
pointer to constant can not be used to change the value being pointed to. Therefore:

```
char ch = â€˜A';
const char *p = &ch;
*p = â€˜B';
```

is not allowed. The program will throw an error.

View All Answers

Question - 21:

What is smart pointer?

Ans:

Smart pointers are objects which store pointers to dynamically allocated (heap) objects. They are like built-in C++ pointers. However, they automatically delete the object pointed to at the appropriate time. They are useful as they ensure proper destruction of dynamically allocated objects (Exceptions). They can also be used to keep track of dynamically allocated objects shared by multiple owners.

They appear as owning the object pointed to and are responsible for deletion of the object when it is no longer needed.
The smart pointer library provides five smart pointer class templates:
scoped_ptr : Simple sole ownership of single objects. Noncopyable.
scoped_array: Simple sole ownership of arrays. Noncopyable.
shared_ptr: Object ownership shared among multiple pointers
shared_array: Array ownership shared among multiple pointers.
weak_ptr: Non-owning observers of an object owned by shared_ptr.
intrusive_ptr: Shared ownership of objects with an embedded reference count.
These templates are designed to complement the std::auto_ptr template.

View All Answers

Question - 22:

What is pointer to member?

Ans:

not to a specific instance of that member in an object. This type of pointer is called a pointer to a class member or a pointer-to-member. It is not same as normal C++ pointer. Instead it provides only an offset into an object of the member's class at which that member can be found. Since member pointers are not true pointers, the . and -> can not be applied to them. Instead we must use the special operators .* and ->* . They allow access to a member of a class.
Example:

```
#include <iostream>
using namespace std;
class MyClass
{
      public:
            int val;
            MyClass(int i)
            {
                  val = i;
            }
            int double_val()
            {
                  return val + val;
            }
};
int main()
{
          int MyClass::*data;         //data member pointer
          int(MyClass::*func)();     //function member pointer
          MyClass obj1(1), obj2(2);       //create objects
          data =&MyClass::val;        //get offset of data val
          func=&MyClass::double_val; //get offset of function double_val()
cout << â€œThe values are:â€•;
cout << ob1.*data << â€œ â€œ <<ob2.*data << â€œ
â€•;
cout<< â€œHere they are doubled:â€•;
cout << (ob1.*func)() << â€œ â€œ <<(ob2.*func)()<< â€œ
â€•;
return 0;
}
```

Here data and func are member pointers. As shown in the program, when declaring pointers to members, you must specify the class and use the scope resolution operator.

View All Answers

Question - 23:

Can you please explain the difference between an inspector and a mutator?

Ans:

An object's state is returned without modifying the object's abstract state by using a function called inspector. Invoking an inspector does not cause any noticeable change in the object's behavior of any of the functions of that object.
A mutator, on the other hand, changes the state of an object which is noticeable by outsiders. It means, it changes the abstract state of the object.
The following code snippet depicts the usage of these two functions:

```
class ShoppingCart {
public:
int addItem(); //Mutator
int numItems() const; //Inspector
};
```

The function addItems() is a mutator. The reason is that it changes the â€˜ShoppingCart' by adding an item.
The function numItems() is an inspector. The reason is that it just updates the count of number of items in the â€˜ShoppingCart'. The const declaration followed by int numItems() specifies that numItems() never change the â€˜ShoppingCart' object..

View All Answers

Question - 24:

When we use this pointer?

Ans:

'this pointer' is used as a pointer to the class object instance by the member function. The address of the class instance is passed as an implicit parameter to the member functions.

View All Answers

Question - 25:

Can you please explain the use of this pointer?

Ans:

When a member function is called, it is automatically passed an implicit argument that is a pointer to the invoking object (ie the object on which the function is invoked). This pointer is known as this pointer. It is internally created at the time of function call.
The this pointer is very important when operators are overloaded.
Example: Consider a class with int and float data members and overloaded Pre-increment operator ++:

```
class MyClass
{
        int i;
        float f;
        public:
            MyClass ()
            {
                    i = 0;
                    f = 0.0;
            }
            MyClass (int x, float y)
            {
                  i = x; f = y;
            }
            MyClass operator ++()
            {
                   i = i + 1;
                   f = f + 1.0;
                 return *this; //this pointer which points to the caller object
            }
            MyClass show()
            {
                        cout<<â€•The elements are:
â€• cout<i<<â€•
<f; //accessing
                                        data members using this
             }
};
int main()
{
          MyClass a;
           ++a; //The overloaded operator function ++()will return a's this
                 pointer
            a.show();
             retun 0;
}
```

The o/p would be:
The elements are:
1
1.0

View All Answers

Question - 26:

Tell me what happens when a pointer is deleted twice?

Ans:

A pointer if not nullified and deleted twice, leads to a trap. If set to null, it wont have much affect if deleted twice.

View All Answers

Question - 27:

What is void pointer using C++?

Ans:

In C++, void represents the absence of type, so void pointers are pointers that point to a value that has no type. The void pointers can point to any data type.
We can declare void pointer as follows.
Void *p;.

View All Answers

Question - 28:

Explain what is NULL pointer and void pointer and what is their use?

Ans:

A NULL pointer has a fixed reserved value that is not zero or space, which indicates that no object is referred. NULL pointers are used in C and C++ as compile-time constant. NULL pointer represents certain conditions, like successor to the last element in linked list, while a consistent structure of the list of nodes are maintained.
A void pointer is a special pointer that points to an unspecified object. A null pointer can not be dereferenced. The address manipulation can directly be done by pointer casting to and from an integral type of sufficient size.

View All Answers

Question - 29:

What is const reference?

Ans:

const references allow you to specify that the data referred to won't be changed. A const reference is actually a reference to const. A reference is inherently const, so when we say const reference, it is not a reference that can not be changed, rather it's a reference to const. Once a reference is bound to refer to an object, it can not be bound to refer to another object. For example:
int &ri = i;
binds ri to refer to i. Then assignment such as:
ri = j;
doesn't bind ri to j. It assigns the value in j to the object referenced by ri, ie i;
This means, if we pass arguments to a function by const references; the function can not change the value stored in those references. This allows us to use const references as a simple and immediate way of improving performance for any function that currently takes objects by value without having to worry that your function might modify the data. The compiler will throw an error if the function tries to modify the value of a const reference.

View All Answers

Question - 30:

What is const pointer?

Ans:

const pointer is a pointer which you don't want to be pointed to a different value. That is, the location stored in the pointer can not change. We can not change where the pointer points. It is declared as:
type * const name
type is data type
name is name of the pointer
eg: char * const p
Since the location to which a const pointer points to can not be changed, the following code:
char ch1 = â€˜A';
char ch2 = â€˜B';
char * const p = &ch1;
p = &ch2;
will throw an error since address stored in p can not be changed.

View All Answers

C++ Most Popular & Related Interview Guides

1 : **C++ Operator Overloading Interview Questions and Answers.**

2 : C++ Exception Handling Interview Questions and Answers.

3 : C++ Template Interview Questions and Answers.

4 : C++ Friend Interview Questions and Answers.

5 : C++ Virtual Functions Interview Questions and Answers.

6 : C++ Constructors Interview Questions and Answers.

7 : C++ Type Checking Interview Questions and Answers.

8 : C++ Inheritance Interview Questions and Answers.

9 : C++ Access Control Interview Questions and Answers.

10 : C++ Inline Function Interview Questions and Answers.

Follow us on FaceBook
www.facebook.com/InterviewQuestionsAnswers.Org

Follow us on Twitter
https://twitter.com/InterviewQA

For any inquiry please do not hesitate to contact us.

Interview Questions Answers.ORG Team
https://InterviewQuestionsAnswers.ORG/
support@InterviewQuestionsAnswers.ORG