Interview question and answer

1. *What is the basic structure of a C program?*

*Ans :- we want a header file and main and opening and close bracket {} to excecute any program*

*#include<stdio.h>*

*{*

*void main()*

*{*

*printf("hello world");*

*}*

*}*

1. *What are the Comments in c and C++?*

*2:-* In computer **programming**, a **comment** is a programmer-readable explanation or annotation in the source code of a computer **program**.

*3. Difference between Declaration and Definition of a variable.*

3: **Declaration** of a variable is for informing to the compiler the following information: name of the variable, type of value it holds and the initial value if any it takes. i.e.,

declaration gives details about the properties of a variable. Whereas,

**Definition** of a variable says where the variable gets stored. i.e.,

memory for the variable is allocated during the definition of the variable

*4: Comment on Local and Global scope of a variable.*

*4: A local variable use or work until we close the bracket in the inside any function in program*

*A global variable will work until the program will not finished.*

*5. When there are a Global variable and Local variable with the same name, how will you access the*

*global variable?*

*5: we use the Scope resolution operator*(::) *to access the global variable*

*6. What is a Constant? Explain with an example.*

*6:* A constant, like a variable, is a memory location where a value can be stored. Unlike variables, constants never change in value. You must initialize a constant when it is created.

Ex-> #include<iostream>

using namespace std;

int main()

{

int a,b,r;

a=10;

b=10;

r=a+b;

cout<<"\n"<<r;

}

7. Comment on Assignment Operator in c and C++.

*7:* Assignment operators are used to assigning value to a variable. The left side operand of the assignment operator is a variable and right side operand of the assignment operator is a value. The value on the right side must be of the same data-type of the variable on the left side otherwise the compiler will raise an error.

*8. What is the difference between equal to (==) and Assignment Operator (=)?*

*8:* **“=”**: This is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left.

“==” is used to compare two values

*9. What are the various Arithmetic Operators in c and C++?*

*9: “+” is used to add two variable (a+b)*

*“-“ is used to subtract two variable (a-b)*

*“\*” is used to multiply two variable(a\*b)*

*“/” is used to divide two variable(a/b=quotient)*

*“%”is a modulus which is used to print remainder*

*10. What are the various Compound Assignment Operators in c and C++?*

*10:*

|  |  |
| --- | --- |
| += | Add the value of the second operand to the value of the first operand; store the result in the object specified by the first operand. |
| –= | Subtract the value of the second operand from the value of the first operand; store the result in the object specified by the first operand. |
| /= | Divide the value of the first operand by the value of the second operand; store the result in the object specified by the first operand. |
| \*= | Multiply the value of the first operand by the value of the second operand; store the result in the object specified by the first operand. |

#include<iostream>

using namespace std;

int main() {

   int a = 3, b = 2;

   a += b;

   cout << a << endl;

   a -= b;

   cout << a << endl;

   a \*= b;

   cout << a << endl;

   a /= b;

   cout << a << endl;

   return 0;

}

*11. difference between Pre and Post Increment/Decrement Operations.*

*11:* **Pre-increment (++i)** − Before assigning the value to the variable, the value is incremented by one.

**Post-increment (i++)** − After assigning the value to the variable, the value is incremented.

**Pre-increment (--i)** − Before assigning the value to the variable, the value is decrement by one.

**Post-increment (i--)** − After assigning the value to the variable, the value is decrement .

*12. What are the Extraction and Insertion operators in C++? Explain with examples.*

12 : C++ is able to input and output the built-in data types using the stream extraction operator >> and the stream insertion operator <<. The stream insertion and stream extraction operators also can be overloaded to perform input and output for user-defined types like an object.

*#include<iostream>*

*using namespace std;*

*int main()*

*{*

*int v;*

*while(v<=10)*

*{*

*v=v+1;*

*cout<<"\n"<<v;*

*}*

*}*

*13. What is the difference between while and do while loop? Explain with examples.*

*13:* In most computer programming languages, a **while loop** is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The **while loop** can be thought of as a repeating if statement.

In most computer programming languages, a **do while loop** is a control flow **statement** that executes a block of code at least once, and then either repeatedly executes the block, or stops executing it, depending on a given boolean condition at the end of the block.

#include<iostream>

using namespace std;

int main()

{ int v;

do

{cout<<"\n"<<v;

v=v+1;

}while(v<=10);}

#include<iostream>

using namespace std;

int main()

{

int v;

while(v<=10)

{

v=v+1;

cout<<"\n"<<v;

}

}

*14. What do you mean by ‘void’ return type?*

14: void is no value is return to main the program is end .

*15. Explain call by Value and call by Reference.*

15 : The **call by value** method of passing arguments to a function copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.

The **call by reference** method of passing arguments to a function copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the **call**.

*16. What are Default Parameters? How are they evaluated in the C++ function?*

16: The default Parameters are used when you provide no arguments or only few arguments while calling a function. The default arguments are used during compilation of program.

#include <iostream>

using namespace std;

int sum(int a, int b=10, int c=20);

int main()

{

cout<<sum(1)<<endl;

cout<<sum(1, 2)<<endl;

cout<<sum(1, 2, 3)<<endl;

return 0;

}

int sum(int a, int b, int c){

int z;

z = a+b+c;

return z;

}

*17. What is the keyword auto for?*

*17:* The **auto** keyword specifies that the type of the variable that is begin declared will automatically be deduced from its initializer and for functions if their return type is **auto** then that will be evaluated by return type expression at runtime.

*18. What is a Static Variable?*

**18:Static variables** in a Function: When a **variable** is declared as **static**, space for it gets allocated for the lifetime of the program. ... So, its value is carried through the function calls. The **variable** count is not getting initialized for every time the function is called.

*19. What are the various Access Specifier in C++?*

*19 There are three Access Specifier in c++ :-*

*1: public 2: protected 3: private*

*20. Explain how functions are classified in C++ ?*

* *20:* [Function with no argument and no return value](https://www.programiz.com/cpp-programming/user-defined-function-types#no_argument_no_return)
* [Function with no argument but return value](https://www.programiz.com/cpp-programming/user-defined-function-types#no_argument_yes_return)
* [Function with argument but no return value](https://www.programiz.com/cpp-programming/user-defined-function-types#yes_argument_no_return)
* [Function with argument and return value](https://www.programiz.com/cpp-programming/user-defined-function-types#yes_argument_yes_return)

*21. What are the basic Data types supported in C Programming Language?*

*21: There are 4 basic data types :-*

*1: int 2: float 3: char 4: long*

*22. What do you mean by the Scope of the variable? What is the scope of the variables in C?*

*22:*

*23. Differentiate between calloc() and malloc()*

*23:*   The malloc() takes a single argument, while

calloc() takes two

Second, malloc() does not initialize the memory allocated,

calloc() initializes the allocated memory to ZERO

*24. What are the valid places where the programmer can apply Break Control Statement?*

*24 only Loop and switch statements where the programmer can apply Break control statements*

*25. Differentiate between Actual Parameters and Formal Parameters.*

*25:* **Actual Parameters are the values that are passed to the function when it is invoked**

**Formal Parameters are the variables defined by the function that receives values when the function is called.**

*26. Can a C program be compiled or executed in the absence of a main()?*

*26:* So the third line “int begin” is replaced by “int **main**” by the preprocessor before the **program** is passed on for the compiler. ... So actually **C program can** never **run** without a **main**() c

*27. What do you mean by a Nested Structure?*

*27* **Nested structure** in C is nothing but **structure** within **structure**. One **structure can** be declared inside other **structure** as **we** declare **structure** members inside a **structure**

*28. What is a C Token?*

*28:* **C Tokens** are the smallest building block or smallest unit of a **C** program. The compiler breaks a program into the smallest possible units and proceeds to the various stages of the compilation, which is called **token**

29. What is Preprocessor?

29: The C preprocessor is a macro preprocessor (allows you to define macros) that transforms your program before it is compiled. These transformations can be the inclusion of header file, macro expansions etc.

All preprocessing directives begin with a # symbol. For example,

#define PI 3.14

30. Why is C called the Mother of all Languages?

30: **C** is often referred to as the **mother of all** programming **language** because it is one of the most popular programming **languages**. Right from the time, it was developed, **C** has become the most widely used and preferred programming **languages**.

31. Mention the features of C Programming Language.

31: Features are :-

1: C is procedure based language

2: C is a high level language

3: C is a case sensitive

4 : C is a Statically type language

32. What is the purpose of printf() and scanf() in C Program?

32 printf()=> this function is used to print the any string or variable from memory to console

scanf()=> this function is used to take the value of the user and store into the specified address

33. What is an array?

33: An **array** is a collection of data items, all of the same type, accessed using a common name. A one-dimensional **array** is like a list; A two dimensional **array** is like a table

34. What is /0 character?

34: A null **character** is a **character** with all its bits set to **zero**. Therefore, it has a numeric value of **zero** and can be used to represent

the end of a string of **characters**, such as a word or phrase.

35. What is the main difference between the Compiler and the Interpreter?

35:

|  |  |
| --- | --- |
| Interpreter | Compiler |
| Translates program one statement at a time. | Scans the entire program and translates it as a whole into machine code. |
| Interpreters usually take less amount of time to analyze the source code. However, the overall execution time is comparatively slower than compilers. | Compilers usually take a large amount of time to analyze the source code. However, the overall execution time is comparatively faster than interpreters. |
| No intermediate object code is generated, hence are memory efficient. | Generates intermediate object code which further requires linking, hence requires more memory. |
| Programming languages like JavaScript, Python, Ruby use interpreters. | Programming languages like C, C++, Java use compilers. |

36. What is Dynamic Memory allocation? Mention the syntax.

36: Programmers can dynamically allocate storage space while the program is running, but programmers cannot create new variable names "on the fly", and for this reason, dynamic allocation requires two criteria:

* Creating the dynamic space in memory
* Storing its address in a pointer (so that space can be accessed

In c++ we use the Syntax (new)n and (delete)

Ex:-

37. What do you mean by Dangling Pointer Variable in C Programming?

37: Sometimes the **programmer** fails to initialize the **pointer** with a valid address, then this type of initialized **pointer** is known as a **dangling pointer** in **C**. **Dangling pointer** occurs at the time of the object destruction when the object is deleted or de-allocated from memory without modifying the value of the **pointer**.

38. Where can we not use &(address operator in C)?

38: In String we not use the &(address operator) Because **string** already stores the address of its variable. So we don't need &. In case of a **string** (character array), the variable itself points to the first element of the array in question. ... Rather for variable with any other type & is **used** to assign the value directly to its address.

39. Write a simple example of a structure in C Language

#include <iostream>

using namespace std;

struct Person

{

char name[50];

int age;

float salary;

};

int main()

{

Person p1;

cout << "Enter Full name: ";

cin.get(p1.name, 50);

cout << "Enter age: ";

cin >> p1.age;

cout << "Enter salary: ";

cin >> p1.salary;

cout << "\nDisplaying Information." << endl;

cout << "Name: " << p1.name << endl;

cout <<"Age: " << p1.age << endl;

cout << "Salary: " << p1.salary;

return 0;

}

40. Differentiate between getch() and getche()

40:It reads a single character from a given input stream and returns the corresponding integer value (typically ASCII value of read character) on success.

Like getch(), this is also a non-standard function present in conio.h. It reads a single character from the keyboard and displays immediately on output screen without waiting for enter key.

41. Which statement is efficient and why? x=x+1; or x++;

41: x++ is more efficient because it will not save the value from right to left it just incremented and pass the value to x

42. Can I declare the same variable name to the variables which have different scopes?

42: Yes — the **scopes** will not overlap so there will be two local **variables**, one per method, each with the **same name**.

43. Mention File operations in C Language.

# 43: BASIC FILE OPERATIONS IN C PROGRAMMING:

There are 4 basic operations that can be performed on any files in C programming language. They are,

1. Opening/Creating a file
2. Closing a file
3. Reading a file
4. Writing in a file

44. What is typecasting?

44 Typecasting is define if want to change the value of the value according to need then forcefully we change the value of the char=> int ,int=>char

45. What are the limitations of scanf() and how can it be avoided?

45: The problems with scanf are (at a minimum):

* using %s to get a string from the user, which leads to the possibility that the string may be longer than your buffer, causing overflow.
* the possibility of a failed scan leaving your file pointer in an indeterminate location.

46. Differentiate between the macros and the functions.

46 **Macros** :- **Macros** are pre-processed which means that all the **macros** would be processed before your program compiles

2:This shows that the **macros** are preprocessed

**Function** :- However, **functions** are not preprocessed but compiled

**2: functions** are not

47. Suppose a global variable and local variable have the same name. Is it is possible to access a global

variable from a block where local variables are defined?

47: A program can **have the same name** for **local** and **global variables** but the value of a **local variable** inside a function will take preference. For **accessing** the **global variable** with **same** rame, you'll **have** to use the scope resolution operator.

48. What is the difference between declaration and definition of a variable/function

48: **Declaration** of a **variable** is for informing to the compiler the following information: name of the **variable**, type of value it holds and the initial value if any it takes.

Whereas, **Definition of a variable** says where the **variable** gets stored.

Difference between declaration and definition function

|  |  |
| --- | --- |
| A variable or a function can be declared any number of times | A variable or a function can be defined only once |
| Memory will not be allocated during declaration | Memory will be allocated |
| int f(int);  The above is a function declaration. This declaration is just for informing the compiler that a function named f with return type and argument as int will be used in the function. | int f(int a)  {  return a;  }  The system allocates memory by seeing the above function definition. |

49. When should we use pointers in a C program?

49:

1. To create dynamic data structures.
2. To pass and handle variable parameters passed to functions.
3. To access information stored in arrays.

50. What is NULL pointer?

50: A Null Pointer is a pointer that does not point to any memory location. It stores the base address of the segment. The null pointer basically stores the Null value while void is the type of the pointer.

A null pointer is a special reserved value which is defined in a **stddef** header file. Here, Null means that the pointer is referring to the 0th memory location.

If we do not have any address which is to be assigned to the pointer, then it is known as a null pointer. When a NULL value is assigned to the pointer, then it is considered as a **Null pointer**.

51. What is Dangling pointer?

51: The most common bugs related to pointers and memory management is dangling/wild pointers. Sometimes the programmer fails to initialize the pointer with a valid address, then this type of initialized pointer is known as a dangling pointer in C.

Dangling pointer occurs at the time of the object destruction when the object is deleted or de-allocated from memory without modifying the value of the pointer.

52. What is variable initialization and why is it important?

52: Variable initialization is define as when we pre define the value in the program and use at any instants in the program. Important because if don’t initialization the variable then the garbage value takes the place

53. Differentiate Source Codes from Object Codes ?

53: **Source code** is generally understood to mean programming statements that are created by a programmer with a text editor or a visual programming tool and then saved in a file.

**Object code** generally refers to the output, a compiled file, which is produced when the **Source Code** is compiled with a C compiler.

54. What is the modulus operator?

54: The modulo operator, denoted by %, is an arithmetic operator. The modulo **division** operator produces the remainder of an integer **division**. produces the remainder when x is divided by y.

55. What is a nested loop?

55: **Nested loop** means a **loop** statement inside another **loop** statement. That is why **nested loops** are also called as “**loop** inside **loop**“

56. Which of the following operators is incorrect and why? ( >=, <=, <>, ==) ?

56:

1. >=” Indicates the operation “greater than or equal to”.
2. “<=” Indicates the operation “less than or equal to”
3. “<>” - It doesn't have any operational meaning. Therefore it is incorrect in the programming languages.
4. “==” This symbol is used to check whether two expressions/numbers are equal or not. Ex: (a==2) this expression implies that it checks whether **a**is equal to 2 or not and proceeds for further operation.

57. How do you declare a variable that will hold string values?

57: The char keyword **can** only **hold** 1 character **value** at a time. By creating an array of characters, you **can** store **string values** in it. Example: "char MyName[50]; " declares a **string variable** named MyName that **can hold** a maximum of 50 characters.

58. Can the curly brackets { } be used to enclose a single line of code?

58: While **curly brackets** are mainly **used** to group several **lines of codes**, it will still work without error if you **used** it for a **single line**.

59. What are header files and what are its uses in C programming?

59:

* All C standard library functions are declared in many header files which are saved as file\_name.h.
* We are including these header files in our C program using “#include <file\_name.h>” command to make use of the functions those are declared in the header files.
* When we include header files in our C program using “#include <filename.h>” command, all C code of the header files are included in C program. Then, this C program is compiled by compiler and executed.

60. What is syntax error?

60: A **syntax error** is an error in the source code of a program. Since computer programs must follow strict syntax to compile correctly,

61. How do you access the values within an array?

61: You can use **array** subscript (or index) to **access** any element stored in **array**. Subscript starts with 0, which means arr[0] represents the first element in the **array** arr.

62. Can two or more operators such as \n and \t be combined in a single line of program code?

62: Yes, it’s perfectly valid to combine operators, especially if the need arises. For example: you can have a code like ”**printf (“Hellonn’World’”)** ” to output the text “Hello” on the first line and “World” enclosed in single quotes to appear on the next two lines

.

63. When is the "void" keyword used in a function?

63: when the function return nothing to program in main then we use the program the void keywords in the function

64. What are compound statements?

64: A **compound statement** (also called a "block") typically appears as the body of another **statement**, such as the if **statement**. Declarations and Types describes the form and meaning of the declarations that can appear at the head of a **compound statement**

65. What is the significance of an algorithm to C programming?

## 65: **Before a program can be written, an algorithm has to be created first. An algorithm provides a step by step procedure on how a solution can be derived. It also acts as a blueprint on how a program will start and end, including what process and computations are involved**

66. What is the advantage of an array over individual variables?

66: 1: We do not have to assign the variable for every value

2: When storing multiple related data, it is a good idea to use arrays. This is because arrays are named using only 1 word followed by an element number. For example: to store the 10 test results of 1 student, one can use 10 different variable names (grade1, grade2, grade3… grade10). With arrays, only 1 name is used, the rest are accessible through the index name (grade[0], grade[1], grade[2]… grade[9]).

67. What is wrong in this statement? scanf("%d",whatnumber);

67: The &(Address) is missing in this syntax .

68. How do you generate random numbers in C?

68: Through randomize() function it is pre declared pre define function in c

Ex #include <stdio.h>  
#include <conio.h>  
#include <stdlib.h>

int main()  
{  
   int n, max, num, c;

   printf("Enter the number of random numbers you want**\n**");  
   scanf("%d", &n);

   printf("Enter the maximum value of random number**\n**");  
   scanf("%d", &max);

   printf("%d random numbers from 0 to %d are:**\n**", n, max);  
     
   randomize();

   for (c = 1; c <= n; c++)  
   {  
      num = random(max);  
      printf("%d**\n**",num);          
   }

   getch();  
   return 0;  
}

69. What does the && operator do in a program code?

69: It is a (&& ) AND operator used in conditional statement and it is work as a true and false in if statement

Ex :- true && true = true

False && false = false

True && false = false

False && true = false

70. In C programming, what command or code can be used to determine if a number of odd or even?

70 (%) modulus operator used to determine the odd and even

71. What are the different types of control structures in programming?

* 71: Sequential: default mode. ...
* Selection: used for decisions, branching -- choosing between 2 or more alternative paths. ...
* Repetition: used for looping, i.e. repeating a piece of code multiple times in a row.

72. What is || operator and how does it function in a program?

72 : It is a (|| ) OR operator used in conditional statement and it is work as a true and false in if statement

Ex :- true || true = true

False || false = false

True || false = true

False || true = true

73. What will be the outcome of the following conditional statement if the value of variable s is 10?

s >=10 && s < 25 && s!=12

73 : The statements in the question is true and it will execute the program reason behind it is all the value which are compare with 10 is true that why the outcome will execute

74. Describe the order of precedence with regards to operators in C.

74: Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.

For example, x = 7 + 3 \* 2; here, x is assigned 13, not 20 because operator \* has a higher precedence than +, so it first gets multiplied with 3\*2 and then adds into 7.

75. How do you determine the length of a string value that was stored in a variable?

75 : we can use the strlen() function for finding the length of the string value explain through the

#include <stdio.h>  
#include <string.h>

int main()  
{  
  char a[100];  
  int length;

  printf("Enter a string to calculate its length**\n**");  
  gets(a);

  length = strlen(a);

  printf("Length of the string = %d**\n**", length);

  return 0;  
}

76. Is it possible to initialize a variable at the time it was declared?

76: yes it is possible to declare the variable at a time it was decalred

Ex :- int n=10; string s=”saksham”;

77. Why is C language being considered a middle level language?

77:- **C** is called **middle**-**level language** because it actually binds the gap between a machine **level language** and high-**level languages**. A user can use **c language** to do System **Programming** (for writing operating system) as well as Application **Programming** (for generating menu driven customer billing system ).

78. What are the different file extensions involved when programming in C?

78: Source codes in C are saved with .C file extension. Header files or library files have the .H file extension. Every time a program source code is successfully compiled, it creates an .OBJ object file, and an executable .EXE file. Then the notation are :- filename.cpp , filename.obj , filename.exe

79. What are reserved words?

79: A **reserved word** is a **word** that cannot be used as an identifier, such as the name of a variable, function, or label – it is "**reserved** from use". This is a syntactic definition, and a **reserved word** may have no meaning. There are a total of 95 **reserved words in C**++.

80. Not all reserved words are written in lowercase. TRUE or FALSE?

80: true in case of c/c++ reserved words are in lowercase and other languages are have some reserves are in capital letter like in java NULL, is used which purely capital

81. What is the difference between the expression "++a" and "a++"?

81 : ++a= means 1st increment and then add in the variable .

a++= means add the values and then increments

82. What would happen to X in this expression: X += 15; (assuming the value of X is 5)

82: x += 5 means

* Find the place identified by x
* Add 5 to it
* Then the value of X is 20;

83. In C language, the variables NAME, name, and Name are all the same. TRUE or FALSE?

83: In all the 3 cases NAME,Name,name the variable is valid to declare

84. What is an endless loop?

84: when the condition in the loop is always true it gets never false in this condition and print the value until the ram get full.

85. What is a program flowchart and how does it help in writing a program?

85:- The **program flowchart** is a diagram that uses a set of standard graphic symbols to represent the sequence of coded instructions fed into a computer, enabling it to perform specified logical and arithmetical operations. It is a great tool to improve **work** efficiency.

86. What is wrong with this program statement? void = 10;

86: it is a syntax error in this statement because void is a reserve word it does not store the value and void itself is null value it also specify that there is nothing retun in the program

87. Is this program statement valid? INT = 10.50;

87: it is syntax error because reserve word does not store the value it provide space in the ram for variable to store the number and in c all reserve words are case sensitive it should look like this ‘int’ and the integer datatypes can store the decimal value but when you execute in the decimal value will not shown

88. When is a "switch" statement preferable over an "if" statement?

88:- **Switch statement** works **better** than multiple **if statements** when you are giving input directly without any **condition** checking in the **statements**. **Switch statement** works well when you want to increase the readability of the code and many alternative available.

89. Is it possible to have a function as a parameter in another function?

89 :-You cannot pass **function** to **another function** as **parameter**. But, you can pass **function** reference to **another function** using **function** pointers. Using **function** pointer you can store reference of a **function** and can pass it to **another function** as **normal** pointer variable.

90. Which function in C can be used to append a string to another string?

90:- (String Concatenation) In the C Programming Language, the strcat function appends a **copy** of the string pointed to by s2 to the end of the string pointed to by s1. It returns a pointer to s1 where the resulting concatenated string resides.

91. What does the characters "r" and "w" mean when writing programs that will make use of files?

91:- **r**" means "read" and **will** open a **file** as input wherein data is to **be** retrieved. "**w**" means "**write**", and **will** open a **file** for output.

92. is it possible to create your own header files?

92 **The** answer to **the** above is yes. **header files** are simply **files** in which you **can** declare **your own** functions that you **can** use in **your** main program or these **can** be used while writing large C programs.

93. In a switch statement, what will happen if a break statement is omitted?

93: Without **break** , the program continues to the next labeled **statement**, executing the **statements** until a **break** or the end of the **statement** is reached. ... **If** there's no default **statement**, and no **case** match is found, none of the **statements** in the **switch** body get executed. There **can** be at most one default **statement**

94. Describe how arrays can be passed to a user defined function

94:- When we pass the address of an array while calling a function then this is called function call by reference. When we pass an address as an argument, the function declaration should have a [pointer](https://beginnersbook.com/2014/01/c-pointers/) as a parameter to receive the passed address

#include <stdio.h>

void disp( int \*num)

{

printf("%d ", \*num);

}

int main()

{

int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};

for (int i=0; i<10; i++)

{

/\* Passing addresses of array elements\*/

disp (&arr[i]);

}

return 0;

}

95. What are pointers?

95:- Some C programming tasks are performed more easily with pointers, and other tasks, such as dynamic memory allocation, cannot be performed without using pointers. So it becomes necessary to learn pointers to become a perfect C programmer. Let's start learning them in simple and easy steps.

96. What is gets() function?

96:- The C library function **char \*gets(char \*str)** reads a line from stdin and stores it into the string pointed to by str. It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.

char \*gets(char \*str)

97. What is the use of a semicolon (;) at the end of every program statement?

97:- (;) it is use to terminate the line in the c++,java,or any other language .

98. What is the wild pointer?

98:- Uninitialized **pointers** are known as **wild pointers** because they point to some arbitrary memory location and may cause a program to crash or behave badly. ... If we want **pointer** to a value (or set of values) without having a variable for the value, we should explicitly allocate memory and put the value in allocated memory.

99. What is void or Generic pointers in C?

99:- A **pointer** to **void** means a **generic pointer** that can point to any data type. ... We can assign the address of any data type to the **void pointer**, and a **void pointer** can be assigned to any type of the **pointer** without performing any explicit typecasting.

100. What is the usage of the pointer in C?

100 The Pointer in C, is a variable that stores address of another variable. A pointer can also be used to refer to another pointer function. A pointer can be incremented/decremented, i.e., to point to the next/ previous **memory** location. The purpose of pointer is to save **memory** space and achieve faster execution time

101. Is that possible to add pointers to each other?

101:- To put it plainly, difference between two pointers give the number of elements of the type that can be stored between the two pointers, but adding them doesn't quite give any meaningful functionality. If there's no meaningful functionality then doesn't it make sense that it is not supported.

102. What is a far pointer in C?

102:- **Far pointer** is a 32-bit **pointer**, can access information which is outside the computer memory in a given segment. To use this **pointer**, one must allocate his/her sector register to store data address in the segment and also another sector register must be stored within the most recent sector.

103. What is a near pointer in C?

103:- Near pointer is a pointer which is used to bit address of up to 16 bits in a given section of the computer memory that is 16 bit enabled. It can only access data of a small size of about 64 kb in a given period, which is the main disadvantage of this.

104. What is the difference between near, far and huge pointers?

## 104**:- Near Pointer**

Near pointer is a pointer which is used to bit address of up to 16 bits in a given section of the computer memory that is 16 bit enabled. It can only access data of a small size of about 64 kb in a given period, which is the main disadvantage of this.

## **Far Pointer**

Far pointer is a 32-bit pointer, can access information which is outside the computer memory in a given segment. To use this pointer, one must allocate his/her sector register to store data address in the segment and also another sector register must be stored within the most recent sector.

## **Huge Pointer**

Huge pointer has the same size of 32-bit to that of a far pointer, and it can also access bits that are located outside the sector. Far pointer which is fixed and hence that part of the sector in which they are located cannot be modified in any way; huge pointers can be.

105. What is the size of a void pointer in C?

105:- The size of a void pointer is **8 bytes**!

106. What is the difference between an uninitialized pointer and a null pointer?

1. 106:- **NULL vs Uninitialized pointer –**An **uninitialized pointer** stores an undefined value.
2. A **null pointer** stores a defined value, but one that is defined by the environment to not be a valid address for any member or object.

107. What is the usage of the NULL pointer in C?

107:- A **null pointer** is a pointer which points nothing.

Some uses of the null pointer are:

a) To initialize a pointer variable when that pointer variable isn’t assigned any valid memory address yet.

b) To pass a null pointer to a function argument when we don’t want to pass any valid memory address.

c) To check for null pointer before accessing any pointer variable. So that, we can perform error handling in pointer related code e.g. dereference pointer variable only if it’s not NULL.

#include <stdio.h>

int main() {

   int \*p= NULL;

   printf("The value of pointer is %u",p);

   return 0;

}

108. What is the FILE pointer?

108:- **File pointer** is a **pointer** which is used to handle and keep track on the files being accessed. A new data type called “**FILE**” is used to declare **file pointer**. ... fopen() function is used to open a **file** that returns a **FILE pointer**. Once **file** is opened, **file pointer** can be used to perform I/O operations on the **file**.

109. What is the advantage of pointers in C?

109 **Advantages of Pointers in C**

**Pointers** provide an efficient way for accessing the elements of an array structure.

**Pointers** are used for dynamic memory allocation as well as deallocation.

**Pointers** are used to form complex data structures such as linked list, graph, tree, etc

110. What is Indirection or Dereference Operator ( \* )?

110 The **dereference operator** or **indirection operator**, sometimes denoted by " \* " (i.e. an asterisk), is a unary **operator** (i.e. one with a single operand) found in C-like languages that include pointer variables. It operates on a pointer variable, and returns an l-value equivalent to the value at the pointer address.

111. What is the address of operator ( &)?

111:- An **address-of operator** is a mechanism within C++ that returns the memory **address** of a variable. These **addresses** returned by the **address-of operator** are known as pointers, because they "point" to the variable in memory. The **address-of operator** is a unary **operator** represented by an ampersand (&).

112. How to declare a pointer to a function in C?

112:- **1)** Unlike normal pointers, a function pointer points to code, not data. Typically a function pointer stores the start of executable code.

**2)**Unlike normal pointers, we do not allocate de-allocate memory using function pointers.

**3)** A function’s name can also be used to get functions’ address. For example, in the below program, we have removed address operator ‘&’ in assignment. We have also changed function call by removing \*, the program still works.

113. What is the difference between pointer and array in C?

113:- **Array** is a collection of variables belongings to the same data type. We can store group of data of same data type in an array.

**Pointer** is a single variable that stores the address of other object/variable.

114. What is an array of pointers?

114 :- An **array of pointers** is an **array** that consists of variables of **pointer** type, which means that the variable is a **pointer** addressing to some other element.

115. What is the return value of malloc (0)?

115:- The **malloc**() function allocates size bytes and **returns** a pointer to the allocated memory. The memory is not initialized. If size is **0**, then **malloc**() **returns** either NULL, or a unique pointer **value** that can later be successfully passed to free().

116. How to access pointer inside the structure in C?

116:- To **access** members of a **structure** using **pointers**, we **use** the -> operator. In this example, the address of person1 is stored in the personPtr **pointer** using personPtr = &person1; . Now, you can **access** the members of person1 using the personPtr **pointer**.

117. How to use a function pointer in structure in C?

117:-

Stuct in C used to represent data structure elemenst, such as student data structure. Struct can contian varible from simple data type and others  from complex ones. complex data type such as varible of function pointer. The easy way to explain the programming ideas by give some simple and suffecient code, Let is start by defining a function pointer and  simple struct.

We define first an function pointer called Operation which return an int value and accepts two integer parameters

typedef int (\*Operation)(int a , int b );

Let us also have a simple struct STR which contains pointer to the Operation function pointer and an integer variable to store the returned value from the Operation variable:

typedef struct \_str {

int result ; // to sotre the resut

Operation opt; // funtion pointer

} STR;