



[Home](#) [Aptitude](#) [Logical](#) [Verbal](#) [CA](#) [Current Affairs](#) [GK](#) [Engineering](#) [Interview](#) [Online Test](#) [Puzzles](#)

Online C Programming Test :: C Programming Test - Random

[Home](#) » [Online Test](#) » [Online C Programming Test](#) » C Programming Test - Random

Hadoop Online Training

Get Hands-on Training on Hadoop ecosystem. Become a Certified Hadoop Developer [edureka.co](#)

Marks : 14/20

Total number of questions	:	20
Number of answered questions	:	20
Number of unanswered questions	:	0

Test Review : View answers and explanation for this test.

1. Is it true that a global variable may have several declarations, but only one definition?

- ☒ A. Yes ✓
☐ B. No ✗

Your Answer: Option A

Correct Answer: Option A

Explanation:

Yes, In all the global variable declarations, you need to use the keyword *extern*.

Learn more problems on : [Declarations and Initializations](#)

Discuss about this problem : [Discuss in Forum](#)

2. Which of the following special symbol allowed in a variable name?

- ☐ A. * (asterisk) ✗
☐ B. | (pipeline) ✗
☐ C. - (hyphen) ✗
☒ D. _ (underscore) ✓

Your Answer: Option D

Correct Answer: Option D

Explanation:

Variable names in C are made up of letters (upper and lower case) and digits. The underscore character ("_") is also permitted. Names must not begin with a digit.

Examples of valid (but not very descriptive) C variable names:

=> foo
=> Bar
=> BAZ
=> foo_bar
=> _foo42
=> _
=> QuUx

Learn more problems on : [Declarations and Initializations](#)

Discuss about this problem : [Discuss in Forum](#)

3. By default a real number is treated as a

- ☐ A. float ❌
- ☒ B. double ✔️
- ☐ C. long double ❌
- ☐ D. far double ❌

Your Answer: Option B

Correct Answer: Option B

Explanation:

In computing, 'real number' often refers to non-complex floating-point numbers. It includes both rational numbers, such as 42 and 3/4, and irrational numbers such as $\pi = 3.14159265\dots$

When the accuracy of the floating point number is insufficient, we can use the *double* to define the number. The *double* is same as *float* but with longer precision and takes double space (8 bytes) than *float*.

To extend the precision further we can use *long double* which occupies 10 bytes of memory space.

Learn more problems on : [Declarations and Initializations](#)

Discuss about this problem : [Discuss in Forum](#)

4. Can we write a function that takes a variable argument list and passes the list to another function?

- ☒ A. Yes ✔️
- ☐ B. No ❌

Your Answer: Option A





Correct Answer: Option A

Learn more problems on : [Variable Number of Arguments](#)

Discuss about this problem : [Discuss in Forum](#)

5. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i = 5;
    while(i-- >= 0)
        printf("%d,", i);
    i = 5;
    printf("\n");
    while(i-- >= 0)
        printf("%i,", i);
    while(i-- >= 0)
        printf("%d,", i);
    return 0;
}
```

- ☐ A. 4, 3, 2, 1, 0, -1 
- ☒ B. 5, 4, 3, 2, 1, 0 
- ☐ C. Error 
- ☐ D. 5, 4, 3, 2, 1, 0 

Your Answer: Option B

Correct Answer: Option A

Explanation:

Step 1: Initially the value of variable *i* is '5'.

Loop 1: *while(i-- >= 0)* here *i* = 5, this statement becomes *while(5-- >= 0)* Hence the *while* condition is satisfied and it prints '4'. (variable '*i*' is decremented by '1'(one) in previous while condition)

Loop 2: *while(i-- >= 0)* here *i* = 4, this statement becomes *while(4-- >= 0)* Hence the *while* condition is satisfied and it prints '3'. (variable '*i*' is decremented by '1'(one) in previous while condition)

Loop 3: *while(i-- >= 0)* here *i* = 3, this statement becomes *while(3-- >= 0)* Hence the *while* condition is satisfied and it prints '2'. (variable '*i*' is decremented by '1'(one) in previous while condition)

Loop 4: *while(i-- >= 0)* here *i* = 2, this statement becomes *while(2-- >= 0)* Hence the *while* condition is satisfied and it prints '1'. (variable '*i*' is decremented by '1'(one) in previous while condition)

Loop 5: *while(i-- >= 0)* here *i* = 1, this statement becomes *while(1-- >= 0)* Hence the *while* condition is satisfied and it prints '0'. (variable '*i*' is decremented by '1'(one) in previous while condition)

Loop 6: *while(i-- >= 0)* here *i* = 0, this statement becomes *while(0-- >= 0)* Hence the *while* condition is satisfied and it prints '-1'. (variable '*i*' is decremented by '1'(one) in previous while condition)

Loop 7: *while(i-- >= 0)* here *i* = -1, this statement becomes *while(-1-- >= 0)* Hence the *while* condition is not satisfied and loop exits.

The output of first while loop is 4,3,2,1,0,-1

Step 2: Then the value of variable *i* is initialized to '5' Then it prints a new line character(\n).

See the above Loop 1 to Loop 7 .

The output of second while loop is 4,3,2,1,0,-1

Step 3: The third while loop, *while(i-- >= 0)* here *i* = -1(because the variable '*i*' is decremented to '-1' by previous while loop and it never initialized.). This statement becomes *while(-1-- >= 0)* Hence the *while* condition is not satisfied and loop exits.

Hence the output of the program is

4,3,2,1,0,-1

4,3,2,1,0,-1

Learn more problems on : [Control Instructions](#)

Discuss about this problem : [Discuss in Forum](#)

6. The '.' operator can be used access structure elements using a structure variable.

- ☒ A.True ✓
☐ B.False ✗

Your Answer: Option A

Correct Answer: Option A

Learn more problems on : [Structures, Unions, Enums](#)

Discuss about this problem : [Discuss in Forum](#)

7.Point out the error in the program.

```
#include<stdio.h>

int main()
{
    const int x;
    x=128;
    printf("%d\n", x);
    return 0;
}
```

- ☐ A.Error: unknown data type *const int* ✗
☒ B.Error: *const* variable have been initialised when declared. ✓
☐ C.Error: stack overflow in x ✗
☐ D.No error ✗

Your Answer: Option B

Correct Answer: Option B

Explanation:

A *const* variable has to be initialized when it is declared. later assigning the value to the const variable will result in an error "Cannot modify the const object".

Hence Option B is correct

Learn more problems on : [Const](#)

Discuss about this problem : [Discuss in Forum](#)

8. What will be the output of the program?

```
#include<stdio.h>
int fun(int, int);
typedef int (*pf) (int, int);
int proc(pf, int, int);

int main()
{
    printf("%d\n", proc(fun, 6, 6));
    return 0;
}
```

```
int fun(int a, int b)
{
    return (a==b);
}
int proc(pf p, int a, int b)
{
    return ((*p)(a, b));
}
```

- ☒ A.6 ✗
☐ B.1 ✓
☐ C.0 ✗
☐ D.-1 ✗

Your Answer: Option A

Correct Answer: Option B

Learn more problems on : [Functions](#)

Discuss about this problem : [Discuss in Forum](#)

9. Does this mentioning array name gives the base address in all the contexts?

- ☐ A. Yes ✗
☒ B. No ✓

Your Answer: Option B

Correct Answer: Option B

Explanation:

No, Mentioning the array name in C or C++ gives the base address in all contexts except one.

Syntactically, the compiler treats the array name as a pointer to the first element. You can reference elements using array syntax, $a[n]$, or using pointer syntax, $*(a+n)$, and you can even mix the usages within an expression.

When you pass an array name as a function argument, you are passing the "value of the pointer", which means that you are implicitly passing the array by reference, even though all parameters in functions are "call by value".

Learn more problems on : [Arrays](#)

Discuss about this problem : [Discuss in Forum](#)

10. Will the following declaration work?

```
typedef struct s
{
    int a;
    float b;
}s;
```

- ☒ A. Yes ✓
☐ B. No ✗

Your Answer: Option A

Correct Answer: Option A

Learn more problems on : [Structures, Unions, Enums](#)

Discuss about this problem : [Discuss in Forum](#)

11. How many times the *while* loop will get executed if a *short int* is 2 byte wide?

```
#include<stdio.h>
int main()
{
    int j=1;
    while(j <= 255)
    {
        printf("%c %d\n", j, j);
        j++;
    }
    return 0;
}
```

- ☐ A. Infinite times ✖
- ☒ B. 255 times ✔
- ☐ C. 256 times ✖
- ☐ D. 254 times ✖

Your Answer: Option B

Correct Answer: Option B

Explanation:

The *while(j <= 255)* loop will get executed 255 times. The size short int(2 byte wide) does not affect the *while()* loop.

Learn more problems on : [Control Instructions](#)

Discuss about this problem : [Discuss in Forum](#)

12. What will be the output of the program?

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

int main()
{
    char *s;
    char *fun();
    s = fun();
    printf("%s\n", s);
    return 0;
}
char *fun()
{
    char buffer[30];
    strcpy(buffer, "RAM");
    return (buffer);
}
```

- ☐ A. 0xffff ✖
- ☐ B. Garbage value ✔
- ☐ C. 0xffee ✖
- ☒ D. Error ✖

Your Answer: Option D

Correct Answer: Option B

Explanation:

The output is unpredictable since *buffer* is an auto array and will die when the control go back to *main*. Thus *s* will be pointing to an array , which not exists.

Learn more problems on : [Memory Allocation](#)

Discuss about this problem : [Discuss in Forum](#)

13.What is *stderr* ?

- ☐ A.standard error ✖
- ☐ B.standard error types ✖
- ☒ C.standard error streams ✔
- ☐ D.standard error definitions ✖

Your Answer: Option C

Correct Answer: Option C

Explanation:

The standard error(stderr) stream is the default destination for error messages and other diagnostic warnings. Like stdout, it is usually also directed to the output device of the standard console (generally, the screen).

Learn more problems on : [Library Functions](#)

Discuss about this problem : [Discuss in Forum](#)

14. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    printf(5+"IndiaBIX\n");
    return 0;
}
```

- ☐ A.Error ✖
- ☐ B.IndiaBIX ✖
- ☒ C.BIX ✔
- ☐ D.None of above ✖

Your Answer: Option C

Correct Answer: Option C

Explanation:

printf(5+"IndiaBIX\n"); In the printf statement, it skips the first 5 characters and it prints "BIX"

Learn more problems on : [Strings](#)

Discuss about this problem : [Discuss in Forum](#)

15.What will be the output of the program ?

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

int main()
{
    char str1[5], str2[5];
    int i;
    gets(str1);
    gets(str2);
    i = strcmp(str1, str2);
    printf("%d\n", i);
    return 0;
}
```

- ☒ A. Unpredictable integer value ✓
- ☐ B. 0 ✗
- ☐ C. -1 ✗
- ☐ D. Error ✗

Your Answer: Option A

Correct Answer: Option A

Explanation:

gets() gets collects a string of characters terminated by a new line from the standard input stream *stdin*.

The *gets(str1)* read the input string from user and store in variable *str1*.

The *gets(str2)* read the input string from user and store in variable *str2*.

The code *i = strcmp(str1, str2);* The *strcmp* not only returns -1, 0 and +1, but also other negative or positive values. So the value of *i* is "unpredictable integer value".

printf("%d\n", i); It prints the value of variable *i*.

Learn more problems on : [Strings](#)

Discuss about this problem : [Discuss in Forum](#)

16. What will be the output of the program (sample.c) given below if it is executed from the command line?

cmd> *sample Jan Feb Mar*

```
/* sample.c */
#include<stdio.h>
#include<dos.h>

int main(int argc, char *argv[])
{
    int i;
    for(i=1; i<_argc; i++)
        printf("%s ", _argv[i]);
    return 0;
}
```

- ☐ A. No output ✗
- ☐ B. sample Jan Feb Mar ✗
- ☐ C. Jan Feb Mar ✓
- ☒ D. Error ✗

Your Answer: Option D

Correct Answer: Option C

Learn more problems on : [Command Line Arguments](#)

Discuss about this problem : [Discuss in Forum](#)

17. What will be the output of the program (sample.c) given below if it is executed from the command line (Turbo C in DOS)?

cmd> sample 1 2 3

```
/* sample.c */
#include<stdio.h>

int main(int argc, char *argv[])
{
    int j;
    j = argv[1] + argv[2] + argv[3];
    printf("%d", j);
    return 0;
}
```

- ☐ A. 6 ✖
- ☐ B. sample 6 ✖
- ☒ C. Error ✔
- ☐ D. Garbage value ✖

Your Answer: Option C

Correct Answer: Option C

Explanation:

Here *argv[1]*, *argv[2]* and *argv[3]* are string type. We have to convert the string to integer type before perform arithmetic operation.

Example: $j = \text{atoi}(\text{argv}[1]) + \text{atoi}(\text{argv}[2]) + \text{atoi}(\text{argv}[3]);$

Learn more problems on : [Command Line Arguments](#)

Discuss about this problem : [Discuss in Forum](#)

18. Point out the error, if any in the *for* loop.

```
#include<stdio.h>
int main()
{
    int i=1;
    for(;;)
    {
        printf("%d\n", i++);
        if(i>10)
            break;
    }
    return 0;
}
```

- ☐ A. There should be a condition in the *for* loop ✖
- ☐ B. The two semicolons should be dropped ✖
- ☐ C. The *for* loop should be replaced with *while* loop. ✖
- ☒ D. No error ✔

Your Answer: Option D

Correct Answer: Option D

Explanation:

Step 1: *for(;;)* this statement will generate infinite loop.

Step 2: *printf("%d\n", i++);* this statement will print the value of variable *i* and increment *i* by 1(one).

Step 3: *if(i>10)* here, if the variable *i* value is greater than 10, then the *for* loop breaks.

Hence the output of the program is

1
2
3
4
5
6
7
8
9
10

Learn more problems on : [Control Instructions](#)

Discuss about this problem : [Discuss in Forum](#)

19.Point out the error in the program?

```
#include<stdio.h>
#include<string.h>
void modify(struct emp*);
struct emp
{
    char name[20];
    int age;
};
int main()
{
    struct emp e = {"Sanjay", 35};
    modify(&e);
    printf("%s %d", e.name, e.age);
    return 0;
}
void modify(struct emp *p)
{
    p ->age=p->age+2;
}
```

- ☐ A.Error: in structure ❌
- ☐ B.Error: in prototype declaration unknown struct emp ✅
- ☒ C.No error ❌
- ☐ D.None of above ❌

Your Answer: Option C

Correct Answer: Option B

Explanation:

The *struct emp* is mentioned in the prototype of the function *modify()* before declaring the structure.To solve this problem declare *struct emp* before the *modify()* prototype.

Learn more problems on : [Structures, Unions, Enums](#)

Discuss about this problem : [Discuss in Forum](#)

20. What will be the output of the program?

```
#include<stdio.h>
int main()
{
    float a=0.7;
    if(a < 0.7)
        printf("C\n");
    else
        printf("C++\n");
    return 0;
}
```

- ☐ A. C ✓
- ☒ B. C++ ✗
- ☐ C. Compiler error ✗
- ☐ D. Non of above ✗

Your Answer: Option B

Correct Answer: Option A

Explanation:

if(a < 0.7) here *a* is a float variable and *0.7* is a double constant. The float variable *a* is less than double constant *0.7*. Hence the *if* condition is satisfied and it prints 'C'

Example:

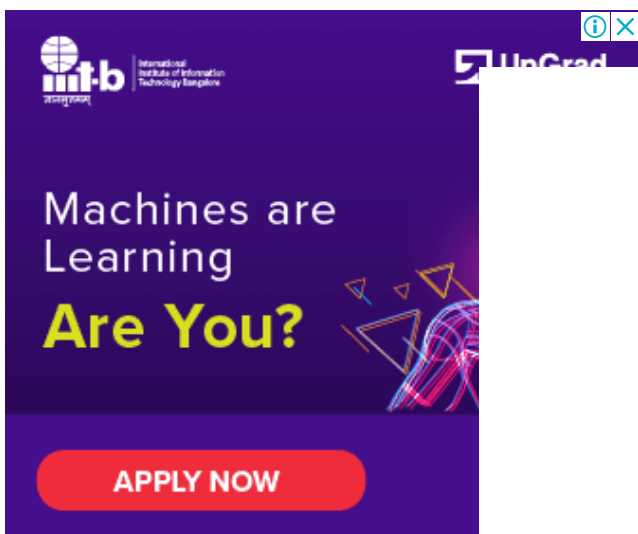
```
#include<stdio.h>
int main()
{
    float a=0.7;
    printf("%.10f %.10f\n",0.7, a);
    return 0;
}
```

Output:

0.7000000000 0.69999999881

Learn more problems on : [Floating Point Issues](#)

Discuss about this problem : [Discuss in Forum](#)





Become Blockchain Expert

Ad TalentSprint



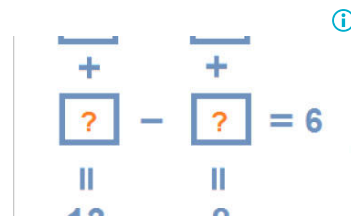
Wipro Placement Papers

indiabix.com



DCDC Converter - MornSun Technologies©

Ad mornsun-power.com



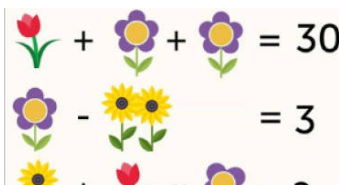
Strings Yes / No Questions - C Programming...

indiabix.com



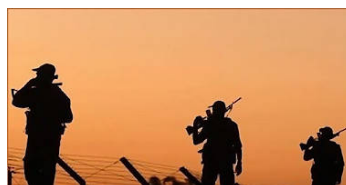
Food Safety Certification

Ad foodsafetyexchange.com



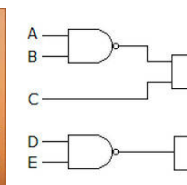
Functions - C Programming Questions and...

indiabix.com



Aptitude Questions and Answers

indiabix.com



Digital Electronics and Communication

indiabix.com

*** END OF THE TEST ***

Feedback

Quality of the Test

:

-- Select -- ▼

Difficulty of the Test

:

-- Select -- ▼

Comments:

...

Submit Feedback

[Current Affairs 2018](#)

[Interview Questions and Answers](#)



© 2009 - 2018 by IndiaBIX™ Technologies. All Rights Reserved. | [Copyright](#) | [Terms of Use & Privacy Policy](#)

Contact us: info-@-@indiab@ix.com [Follow us on twitter!](#)