

Technical Interview Questions
[C++ Interview Questions](#)
[C# Interview Questions](#)
[C Interview Questions](#)

[.....More](#)

Aptitude Interview Questions
[C ++ Aptitude Questions](#)
[C Aptitude Questions 2](#)

[.....More](#)

Tutorials

[C Tutorial](#)
[C++ Tutorial](#)

[.....More](#)

Programming Source Codes

[C/C++ Source Codes](#)
[C# Source Codes](#)

[.....More](#)

Soft Skills

[Communication Skills](#)
[Leadership Skills](#)

[.....More](#)

Top of Form

1470006299	
techpreparation	http://techprepara
	Interview
1	from
0	

Subscribe to our Newsletters

Name:

Email:

Bottom of Form

Top of Form

<input type="text"/>		<input type="button" value="Google Search"/>
pub-7003540323	1	3422992112
ISO-8859-1	ISO-8859-1	active
GALT:#008000;G		en

Bottom of Form

C Aptitude Questions and Answers

Predict the output or error(s) for the following:

```
1. void main()
{
    int const * p=5;
    printf("%d",++(*p));
}
```

Answer:

Compiler error: Cannot modify a constant value.

Explanation:

p is a pointer to a "constant integer". But we tried to change the value of the "constant integer".

```
2. main()
{
    char s[ ]="man";
    int i;
    for(i=0;s[ i ];i++)
        printf("\n%c%c%c%c",s[
i ],*(s+i),*(i+s),i[s]);
}
```

Answer:

```
mmmm
aaaa
nnnn
```

Explanation:

s[i], *(i+s), *(s+i), i[s] are all different ways of expressing the same idea. Generally array name is the base address for that array. Here s is the base address. i is the index number/displacement from the base

address. So, indirecting it with * is same as s[i]. i[s] may be surprising. But in the case of C it is same as s[i].

```

3.  main()
{
    float me = 1.1;
    double you = 1.1;
    if(me==you)
printf("I love U");
else
    printf("I hate U");
}

```

Answer:
I hate U

Explanation:

For floating point numbers (float, double, long double) the values cannot be predicted exactly. Depending on the number of bytes, the precision with of the value represented varies. Float takes 4 bytes and long double takes 10 bytes. So float stores 0.9 with less precision than long double.

Rule of Thumb:

Never compare or at-least be cautious when using floating point numbers with relational operators (== , >, <, <=, >=, !=) .

```

4.  main()
{
    static int var = 5;
    printf("%d ",var--);
    if(var)
        main();
}

```

Answer:
5 4 3 2 1

Explanation:

When static storage class is given, it is initialized once. The change in the value of a static variable is retained even between the function calls. Main is also treated like any other ordinary function, which can be called recursively.

```

5.  main()
{
    int c[ ]={2.8,3.4,4,6.7,5};
    int j,*p=c,*q=c;
    for(j=0;j<5;j++) {
        printf(" %d ",*c);
        ++q;    }
    for(j=0;j<5;j++){

```

```
printf(" %d ",*p);  
++p;  }  
}
```

Answer:

2 2 2 2 2 2 3 4 6 5

Explanation:

Initially pointer c is assigned to both p and q. In the first loop, since only q is incremented and not c , the value 2 will be printed 5 times. In second loop p itself is incremented. So the values 2 3 4 6 5 will be printed.

```
6.  main()  
{  
    extern int i;  
    i=20;  
    printf("%d",i);  
}
```

Answer:

Linker Error : Undefined symbol '_i'

Explanation:

extern storage class in the following declaration,

```
extern int i;
```

specifies to the compiler that the memory for i is allocated in some other program and that address will be given to the current program at the time of linking. But linker finds that no other variable of name i is available in any other program with memory space allocated for it. Hence a linker error has occurred .

Page Numbers : 1 [2](#) [3](#) [4](#) [5](#)
[6](#) [7](#) [8](#)



Have a Question ? [post your questions](#) here. It will be answered as soon as possible.

Check [C Aptitude Questions](#) for more C Aptitude Interview Questions with Answers

Check [Placement Papers](#) for more IT Companies Paper

Technical Interview Questions

[C++ Interview Questions](#)

[C# Interview Questions](#)

[C Interview Questions](#)

[.....More](#)

Aptitude Interview Questions

[C++ Aptitude Questions](#)

[C Aptitude Questions 2](#)

[.....More](#)

Tutorials

[C Tutorial](#)

[C++ Tutorial](#)

[.....More](#)

Programming Source Codes

[C/C++ Source Codes](#)

[C# Source Codes](#)

[.....More](#)

Soft Skills

[Communication Skills](#)

[Leadership Skills](#)

[.....More](#)

Top of Form

1470006299

techpreparation

http://techprepara

Interview

1

from

0

Subscribe to our Newsletters

Name:

Email:

Bottom of Form



Top of Form

pub-7003540323

1

3422992112

ISO-8859-1

ISO-8859-1

active

GALT:#008000;G

en

Bottom of Form

C Aptitude Questions and Answers

Predict the output or error(s) for the following:

7. main()

```

{
    int i=-1,j=-1,k=0,l=2,m;
    m=i++&& j++&& k++||l++;
    printf("%d %d %d %d %d",i,j,k,l,m);
}

```

Answer:

0 0 1 3 1

Explanation :

Logical operations always give a result of 1 or 0 . And also the logical AND (&&) operator has higher priority over the logical OR (||) operator. So the expression 'i++ && j++ && k++' is executed first. The result of this expression is 0 (-1 && -1 && 0 = 0). Now the expression is 0 || 2 which evaluates to 1 (because OR operator always gives 1 except for '0 || 0' combination- for which it gives 0). So the value of m is 1. The values of other variables are also incremented by 1.

8. main()

```

{
    char *p;
    printf("%d %d ",sizeof(*p),sizeof(p));
}

```

Answer:

1 2

Explanation:

The sizeof() operator gives the number of bytes taken by its operand. P is a character pointer, which needs one byte for storing its value (a character). Hence sizeof(*p) gives a value of 1. Since it needs two bytes to store the address of the character pointer sizeof(p) gives 2.

```

9.  main()
{
    int i=3;
    switch(i)
    {
        default:printf("zero");
        case 1: printf("one");
                break;
        case 2:printf("two");
                break;
        case 3: printf("three");
                break;
    }
}

```

Answer :
three

Explanation :
The default case can be placed anywhere inside the loop. It is executed only when all other cases doesn't match.

```

10.  main()
{
    printf("%x",-1<<4);
}

```

Answer:
fff0

Explanation :
-1 is internally represented as all 1's. When left shifted four times the least significant 4 bits are filled with 0's. The %x format specifier specifies that the integer value be printed as a hexadecimal value.

```

11.  main()
{
    char string[]="Hello World";
    display(string);
}
void display(char *string)
{
    printf("%s",string);
}

```

Answer:
Compiler Error : Type mismatch in redeclaration of function display

Explanation :
In third line, when the function display is encountered,

the compiler doesn't know anything about the function display. It assumes the arguments and return types to be integers, (which is the default type). When it sees the actual function display, the arguments and type contradicts with what it has assumed previously. Hence a compile time error occurs.

```
12.    main()  
{  
    int c=- -2;  
    printf("c=%d",c);  
}
```

Answer:

c=2;

Explanation:

Here unary minus (or negation) operator is used twice. Same maths rules applies, ie. minus * minus= plus.

Note:

However you cannot give like --2. Because -- operator can only be applied to variables as a decrement operator (eg., i--). 2 is a constant and not a variable.

```
13.    #define int char  
main()  
{  
    int i=65;  
    printf("sizeof(i)=%d",sizeof(i));  
}
```

Answer:

sizeof(i)=1

Explanation:

Since the #define replaces the string int by the macro char

```
14.    main()  
{  
    int i=10;  
    i=!i>14;  
    Printf ("i=%d",i);  
}
```

Answer:

i=0

Explanation:

In the expression !i>14 , NOT (!) operator has more precedence than ' >' symbol. ! is a unary logical operator. !i (!10) is 0 (not of true is false). 0>14 is false (zero).

Page Numbers : [1](#) [2](#) [3](#) [4](#) [5](#)
[6](#) [7](#) [8](#)



Have a Question ? [post your questions](#) here. It will be answered as soon as possible.

Check [C Aptitude Questions](#) for more C Aptitude Interview Questions with Answers

Check [Placement Papers](#) for more IT Companies Paper

Bookmark & Share

X

Select from these web-based feed readers:

AOL
Bloglines
Google Reader
My MSN
Netvibes
Newsisfree
Pageflakes
Technorati
Yahoo

No matching services.

.netShoutout
100zakladok
A1-Webmarks
Adifni
Aero
AIM Share
Amazon
Amen Me!
AOL Mail
Arto
Ask
Aviary Capture
Baidu
Bebo
Bit.ly
BizSugar
Bleetbox
Blinklist
Blip
Blogger
Bloggy
Blogmarks
Bobrdobr
BonzoBox
Bordom

Box.net
Brainify
Bryderi.se
BuddyMarks
Buzz
Camyoo
Care2
Cirip
CiteULike
ClassicalPlace
Clickazoo
Colivia.de
Connotea
COSMiQ
Delicious
DesignBump
Designmoo
Digg
Diggita
Diglog
Digo
Diigo
Dipdive
DoMelhor
Doower
Dosti
DotNetKicks
Dropjack
Dzone
Edelight
eKudos
eLert Gadget
Email
Embarkons
euCliquei
Evernote
Fabulously40
Facebook
Fark
Farkinda
FAVable
Faves
Favorites
Favoritus
Flaker
Floss.pro
Fnews
Folkd
Fresqui
FriendFeed
Friendster
funP
fwisp
Gabbr

Gacetilla
GamesN
GlobalGrind
GluvSnap
Gmail
Google
Google Reader
Gravee
Grumper
Haber.gen.tr
Hacker News
Hadash Hot
Hatena
Hazarkor
Hedgehogs.net
HelloTxt
HEMiDEMi
Hipstr
Hitmarks
Hot Bookmark
Hotklix
Hotmail
HTML Validator
Hyves
Identi.ca
Instapaper
InvestorLinks
Jamespot
Jumptags
Kaboodle
Kaezur
KiRTSY
Kledy
koornk
Kudos
Laaikit
Librerio
Link Ninja
Link-a-Gogo
LinkedIn
Linkuj.cz
Live
Livefavoris
LiveJournal
Lunch.com
Lynki
Meccho
meinVZ
Memori.ru
Menéame
Mindbodygreen
Mister Wong
Mixx
Multiply

myAOL
Mylinkvault
MySpace
N4G
NetLog
Netvibes
Netvouz
NewsTrust
Newsvine
Nujj
OKNOtizie
Oneview
Orkut
Osmosus
Oyyla
PDF Online
PhoneFavs
PimpThisBlog
Ping.fm
Planypus
Plaxo
Plurk
Polladium
Posterous
Print
PrintFriendly
Propeller
Pusha
Quantcast
Read It Later
Reddit
Scoop.at
Segnalo
Sekoman
Shaveh
She Told Me
Simpy
Slashdot
Smak News
SodaHead
Sonico
Speedtile
Sphinn
springpad
Spruzer
Squidoo
Startaid
Startlap
Strands
studiVZ
Stuffpit
StumbleUpon
Stumpedia
Stylehive

- Surfpeople
- Svejo
- Symbaloo
- Tagza
- Technorati
- TellMyPolitician
- ThisNext
- Tip'd
- Transferr
- Translate
- Tuling
- Tumblr
- Tusul
- TweetMeme
- Twitter
- Typepad
- Viadeo
- Virb
- Vyoom
- Webnews
- Whois Lookup
- Windy Citizen
- WireFan
- WordPress
- Worio
- Wovre
- Wykop
- Y! Bookmarks
- Y! Mail
- Yammer
- Yardbarker
- Yazzem
- Yigg
- Yoolink
- Yorumcuyum
- Youbookmarks
- YouMob
- [Suggest a service »](#)

Done

Message sent! [Share again.](#)



AddThis for Internet Explorer^{NEW}

Bookmark, email or share any page, anytime.

[Install](#)

Top of Form

▲

▼

◀

▶

To: (email address)

From: (email address)

Note: (optional)

255 character limit

Bottom of Form

[Get the AddThis Toolbar](#)[Privacy](#)[AddThis](#)

Bookmark & Share

Make sharing easier with AddThis for Internet Explorer.



To:

From:

From:

Note:

- Email
- Favorites
- Print
- Delicious
- Digg
- Google
- MySpace
- Live
- Facebook
- StumbleUpon
- Twitter
- More... (224)

[Get the AddThis Toolbar](#)[AddThis](#) **Predict the output or error(s) for the following:**

```
15. #include
main()
{
char s[]={'a','b','c','\n','c','\0'};
char *p,*str,*str1;
p=&s[3];
str=p;
str1=s;
```

```
printf("%d",++*p + ++*str1-32);  
}
```

Answer:
77

Explanation:

p is pointing to character '\n'. str1 is pointing to character 'a' ++*p. "p is pointing to '\n' and that is incremented by one." the ASCII value of '\n' is 10, which is then incremented to 11. The value of ++*p is 11. ++*str1, str1 is pointing to 'a' that is incremented by 1 and it becomes 'b'. ASCII value of 'b' is 98.

Now performing (11 + 98 - 32), we get 77("M");

So we get the output 77 :: "M" (Ascii is 77).

```
16. #include  
main()  
{  
int a[2][2][2] = { {10,2,3,4}, {5,6,7,8} };  
int *p,*q;  
p=&a[2][2][2];  
*q=**a;  
printf("%d----%d",*p,*q);  
}
```

Answer:
SomeGarbageValue---1

Explanation:

p=&a[2][2][2] you declare only two 2D arrays, but you are trying to access the third 2D(which you are not declared) it will print garbage values. *q=**a starting address of a is assigned integer pointer. Now q is pointing to starting address of a. If you print *q, it will print first element of 3D array.

```
17. #include  
main()  
{  
struct xx  
{  
    int x=3;  
    char name[]="hello";  
};  
struct xx *s;  
printf("%d",s->x);  
printf("%s",s->name);  
}
```

Answer:
Compiler Error

Explanation:

You should not initialize variables in declaration

```
18. #include  
main()
```

```

{
struct xx
{
int x;
struct yy
{
char s;
         struct xx *p;
};
struct yy *q;
};
}

```

Answer:
Compiler Error

Explanation:

The structure yy is nested within structure xx. Hence, the elements of yy are to be accessed through the instance of structure xx, which needs an instance of yy to be known. If the instance is created after defining the structure the compiler will not know about the instance relative to xx. Hence for nested structure yy you have to declare member.

```

19. main()
{
printf("\nab");
printf("\bsi");
printf("\rha");
}

```

Answer:
hai

Explanation:

\n - newline
\b - backspace
\r - linefeed

```

20. main()
{
int i=5;
printf("%d%d%d%d%d",i++,i--,++i,--i,i);
}

```

Answer:
45545

Explanation:

The arguments in a function call are pushed into the stack from left to right. The evaluation is by popping out from the stack. and the evaluation is from right to left, hence the result.

```

21. #define square(x) x*x
main()

```

```

{
int i;
i = 64/square(4);
printf("%d",i);
}

```

Answer:
64

Explanation:
the macro call square(4) will substituted by 4*4 so the expression becomes $i = 64/4*4$.
Since / and * has equal priority the expression will be evaluated as $(64/4)*4$ i.e. $16*4 = 64$

```

22. main()
{
char *p="hai friends",*p1;
p1=p;
while(*p!='\0') ++*p++;
printf("%s %s",p,p1);
}

```

Answer:
ibj!gsjfoet

Explanation:
++*p++ will be parse in the given order
Ø *p that is value at the location currently pointed by p will be taken
Ø ++*p the retrieved value will be incremented
Ø when ; is encountered the location will be incremented that is p++ will be executed
Hence, in the while loop initial value pointed by p is 'h', which is changed to 'i' by executing ++*p and pointer moves to point, 'a' which is similarly changed to 'b' and so on. Similarly blank space is converted to '!'. Thus, we obtain value in p becomes "ibj!gsjfoet" and since p reaches '\0' and p1 points to p thus p1 doesnot print anything.

```

23. #include
#define a 10
main()
{
#define a 50
printf("%d",a);
}

```

Answer:
50

Explanation:
The preprocessor directives can be redefined anywhere in the program. So the most recently assigned value will be taken.

```

24. #define clrscr() 100
main()
{
clrscr();
}

```

```
printf("%d\n",clrscr());  
}
```

Answer:
100

Explanation:

Preprocessor executes as a separate pass before the execution of the compiler. So textual replacement of clrscr() to 100 occurs. The input program to compiler looks like this :

```
main()  
{  
    100;  
    printf("%d\n",100);  
}
```

Note:

100; is an executable statement but with no action. So it doesn't give any problem

Predict the output or error(s) for the following:

25. main()

```
{  
printf("%p",main);  
}
```

Answer:

Some address will be printed.

Explanation:

Function names are just addresses (just like array names are addresses). main() is also a function. So the address of function main will be printed. %p in printf specifies that the argument is an address. They are printed as hexadecimal numbers.

26. main()

```
{  
clrscr();  
}  
clrscr();
```

Answer:

No output/error

Explanation:

The first clrscr() occurs inside a function. So it becomes a function call. In the second clrscr(); is a function declaration (because it is not inside any function).

27. enum colors {BLACK,BLUE,GREEN}

main()

```
{  
  
printf("%d..%d..%d",BLACK,BLUE,GREEN);  
  
return(1);  
}
```

Answer:
0..1..2

Explanation:
enum assigns numbers starting from 0, if not explicitly defined.

```
28. void main()  
{  
  char far *farther,*farthest;  
  
  printf("%d..%d",sizeof(farther),sizeof(farthest));  
  
}
```

Answer:
4..2

Explanation:
the second pointer is of char type and not a far pointer

```
29. main()  
{  
  int i=400,j=300;  
  printf("%d..%d");  
}
```

Answer:
400..300

Explanation:
printf takes the values of the first two assignments of the program. Any number of printf's may be given. All of them take only the first two values. If more number of assignments given in the program, then printf will take garbage values.

```
30. main()  
{  
  char *p;  
  p="Hello";  
  printf("%c\n",&*p);  
}
```

Answer:
H

Explanation:
* is a dereference operator & is a reference operator. They can be applied any number of times provided it is meaningful. Here p points to the first character in the string "Hello". *p dereferences it and so its value is H. Again & references it to an address and * dereferences it to the value H.

```
31. main()  
{  
  int i=1;  
  while (i<=5)  
  {
```

```

    printf("%d",i);
    if (i>2)
        goto here;
    i++;
}
}
fun()
{
    here:
    printf("PP");
}

```

Answer:

Compiler error: Undefined label 'here' in function main

Explanation:

Labels have functions scope, in other words The scope of the labels is limited to functions . The label 'here' is available in function fun() Hence it is not visible in function main.

```

32.    main()
{
    static char names[5][20]= {"pascal","ada","cobol","fortran","perl"};
    int i;
    char *t;
    t=names[3];
    names[3]=names[4];
    names[4]=t;
    for (i=0;i<=4;i++)
        printf("%s",names[i]);
}

```

Answer:

Compiler error: Lvalue required in function main

Explanation:

Array names are pointer constants. So it cannot be modified.

```

33.    void main()
{
    int i=5;
    printf("%d",i++ + ++i);
}

```

Answer:

Output Cannot be predicted exactly.

Explanation:

Side effects are involved in the evaluation of i

```

34.    void main()
{
    int i=5;
    printf("%d",i+++++i);
}

```

```
}
```

Answer:

Compiler Error

Explanation:

The expression i+++++i is parsed as i ++ ++ + i which is an illegal combination of operators.

```
35. #include  
main()  
{  
int i=1,j=2;  
switch(i)  
  {  
  case 1: printf("GOOD");  
    break;  
  case j: printf("BAD");  
    break;  
  }  
}
```

Answer:

Compiler Error: Constant expression required in function main.

Explanation:

The case statement can have only constant expressions (this implies that we cannot use variable names directly so an error).

Note:

Enumerated types can be used in case statements.

Predict the output or error(s) for the following:

```
36. main()  
{  
int i;  
printf("%d",scanf("%d",&i)); // value 10 is given as input here  
}
```

Answer:

1

Explanation:

Scanf returns number of items successfully read and not 1/0. Here 10 is given as input which should have been scanned successfully. So number of items read is 1.

```
37. #define f(g,g2) g##g2  
main()  
{  
int var12=100;  
printf("%d",f(var,12));  
  }
```

Answer:

100

```

38.   main()
{
int i=0;

for(;i++;printf("%d",i)) ;
printf("%d",i);
}

```

Answer:
1

Explanation:
before entering into the for loop the checking condition is "evaluated". Here it evaluates to 0 (false) and comes out of the loop, and i is incremented (note the semicolon after the for loop).

```

39.   #include
main()
{
char s[]={'a','b','c','\n','c','\0'};
char *p,*str,*str1;
p=&s[3];
str=p;
str1=s;
printf("%d",++*p + ++*str1-32);
}

```

Answer:
M

Explanation:
p is pointing to character '\n'.str1 is pointing to character 'a' ++*p meAnswer:"p is pointing to '\n' and that is incremented by one." the ASCII value of '\n' is 10. then it is incremented to 11. the value of ++*p is 11. ++*str1 meAnswer:"str1 is pointing to 'a' that is incremented by 1 and it becomes 'b'. ASCII value of 'b' is 98. both 11 and 98 is added and result is subtracted from 32.
i.e. (11+98-32)=77("M");

```

40.   #include
main()
{
struct xx
{
int x=3;
char name[]="hello";
};
struct xx *s=malloc(sizeof(struct xx));
printf("%d",s->x);
printf("%s",s->name);
}

```

Answer:
Compiler Error

Explanation:

Initialization should not be done for structure members inside the structure declaration

```
41. #include  
main()  
{  
struct xx  
{  
    int x;  
    struct yy  
    {  
        char s;  
        struct xx *p;  
    };  
        struct yy *q;  
};  
}
```

Answer:

Compiler Error

Explanation:

in the end of nested structure yy a member have to be declared

```
42. main()  
{  
extern int i;  
i=20;  
printf("%d",sizeof(i));  
}
```

Answer:

Linker error: undefined symbol '_i'.

Explanation:

extern declaration specifies that the variable i is defined somewhere else. The compiler passes the external variable to be resolved by the linker. So compiler doesn't find an error. During linking the linker searches for the definition of i. Since it is not found the linker flags an error.

```
43. main()  
{  
printf("%d", out);  
}
```

int out=100;

Answer:

Compiler error: undefined symbol out in function main.

Explanation:

The rule is that a variable is available for use from the point of declaration. Even though a is a global variable, it is not available for main. Hence an error

Predict the output or error(s) for the following:

```
44. main()
```

```

{
extern out;
printf("%d", out);
}
int out=100;

```

Answer:
100

Explanation:

This is the correct way of writing the previous program.

45. main()

```

{
show();
}
void show()
{
printf("I'm the greatest");
}

```

Answer:

Compiler error: Type mismatch in redeclaration of show.

Explanation:

When the compiler sees the function show it doesn't know anything about it. So the default return type (ie, int) is assumed. But when compiler sees the actual definition of show mismatch occurs since it is declared as void. Hence the error.

The solutions are as follows:

1. declare void show() in main() .
2. define show() before main().
3. declare extern void show() before the use of show().

46. main()

```

{
int a[2][3][2] = {{ {2,4},{7,8},{3,4}},{ {2,2},{2,3},{3,4} }};
printf("%u %u %u %d \n",a,*a,**a,***a);
printf("%u %u %u %d \n",a+1,*a+1,**a+1,***a+1);
}

```

Answer:

100, 100, 100, 2
114, 104, 102, 3

47. main()

```

{
int a[ ] = {10,20,30,40,50},j,*p;
for(j=0; j<5; j++)
{
printf("%d" ,*a);
a++;
}
p = a;
for(j=0; j<5; j++)

```

```

    {
printf("%d " ,*p);
p++;
    }
}

```

Answer:

Compiler error: lvalue required.

Explanation:

Error is in line with statement `a++`. The operand must be an lvalue and may be of any of scalar type for the any operator, array name only when subscripted is an lvalue. Simply array name is a non-modifiable lvalue.

```

48.    main( )
{
    static int a[ ] = {0,1,2,3,4};
    int *p[ ] = {a,a+1,a+2,a+3,a+4};
    int **ptr = p;
    ptr++;
    printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
    *ptr++;
    printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
    *++ptr;
    printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
    ++*ptr;
        printf("\n %d %d %d", ptr-p, *ptr-a, **ptr);
}

```

Answer:

```

111
222
333
344

```

```

49.    main( )
{
    void *vp;
    char ch = 'g', *cp = "goofy";
    int j = 20;
    vp = &ch;
    printf("%c", *(char *)vp);
    vp = &j;
    printf("%d",*(int *)vp);
    vp = cp;
    printf("%s",(char *)vp + 3);
}

```

Answer:

```

g20fy

```

Explanation:

Since a void pointer is used it can be type casted to any other type pointer. `vp = &ch` stores address of char `ch` and the next statement prints the value stored in `vp` after type

casting it to the proper data type pointer. the output is 'g'. Similarly the output from second printf is '20'. The third printf statement type casts it to print the string from the 4th value hence the output is 'fy'.

```
50. main ( )
{
  static char *s[ ] = {"black", "white", "yellow", "violet"};
  char **ptr[ ] = {s+3, s+2, s+1, s}, ***p;
  p = ptr;
  **++p;
  printf("%s",*--*++p + 3);
}
```

Answer:

ck

Explanation:

In this problem we have an array of char pointers pointing to start of 4 strings. Then we have ptr which is a pointer to a pointer of type char and a variable p which is a pointer to a pointer to a pointer of type char. p hold the initial value of ptr, i.e. p = s+3. The next statement increment value in p by 1, thus now value of p = s+2. In the printf statement the expression is evaluated *++p causes gets value s+1 then the pre decrement is executed and we get s+1 - 1 = s. the indirection operator now gets the value from the array of s and adds 3 to the starting address. The string is printed starting from this position. Thus, the output is 'ck'.

```
51. main()
{
  int i, n;
  char *x = "girl";
  n = strlen(x);
  *x = x[n];
  for(i=0; i
    {
  printf("%s\n",x);
  x++;
    }
}
```

Answer:

(blank space)

irl

rl

l

Explanation:

Here a string (a pointer to char) is initialized with a value "girl". The strlen function returns the length of the string, thus n has a value 4. The next statement assigns value at the nth location ('\0') to the first location. Now the string becomes "\0irl". Now the printf statement prints the string after each iteration it increments its starting position. Loop starts from 0 to 4. The first time x[0] = '\0' hence it prints nothing and pointer value is incremented. The second time it prints from x[1] i.e. "irl" and the third time it prints "rl" and the last time it prints "l" and the loop terminates.

Predict the output or error(s) for the following:

```
52.  int i,j;
      for(i=0;i<=10;i++)
      {
        j+=5;
        assert(i<5);
      }
```

Answer:

Runtime error: Abnormal program termination.
assert failed (i<5), ,

Explanation:

asserts are used during debugging to make sure that certain conditions are satisfied. If assertion fails, the program will terminate reporting the same. After debugging use,
#undef NDEBUG

and this will disable all the assertions from the source code. Assertion is a good debugging tool to make use of.

```
53.  main()
      {
        int i=-1;
        +i;
        printf("i = %d, +i = %d \n",i,+i);
      }
```

Answer:

i = -1, +i = -1

Explanation:

Unary + is the only dummy operator in C. Where-ever it comes you can just ignore it just because it has no effect in the expressions (hence the name dummy operator).

54. What are the files which are automatically opened when a C file is executed?

Answer:

stdin, stdout, stderr (standard input,standard output,standard error).

55. what will be the position of the file marker?

- a: fseek(ptr,0,SEEK_SET);
- b: fseek(ptr,0,SEEK_CUR);

Answer :

- a: The SEEK_SET sets the file position marker to the starting of the file.
- b: The SEEK_CUR sets the file position marker to the current position of the file.

```
56.  main()
      {
        char name[10],s[12];
        scanf(" %[^\\"]",s);
      }
How scanf will execute?
```

Answer:

First it checks for the leading white space and discards it. Then it matches with a quotation mark and then it reads all character upto another quotation mark.

57. What is the problem with the following code segment?
while ((fgets(receiving array,50,file_ptr)) != EOF)

Answer & Explanation:

fgets returns a pointer. So the correct end of file check is checking for != NULL.

58. main()
{
main();
}

Answer:

Runtime error : Stack overflow.

Explanation:

main function calls itself again and again. Each time the function is called its return address is stored in the call stack. Since there is no condition to terminate the function call, the call stack overflows at runtime. So it terminates the program and results in an error.

59. main()
{
char *cptr,c;
void *vptr,v;
c=10; v=0;
cptr=&c; vptr=&v;
printf("%c%v",c,v);
}

Answer:

Compiler error (at line number 4): size of v is Unknown.

Explanation:

You can create a variable of type void * but not of type void, since void is an empty type. In the second line you are creating variable vptr of type void * and v of type void hence an error.

60. main()
{
char *str1="abcd";
char str2[]="abcd";
printf("%d %d %d",sizeof(str1),sizeof(str2),sizeof("abcd"));
}

Answer:

2 5 5

Explanation:

In first sizeof, str1 is a character pointer so it gives you the size of the pointer variable. In second sizeof the name str2 indicates the name of the array whose size is 5 (including the '\0' termination character). The third sizeof is similar to the second one.

```

61.  main()
      {
      char not;
      not=!2;
      printf("%d",not);
      }

```

Answer:
0

Explanation:

! is a logical operator. In C the value 0 is considered to be the boolean value FALSE, and any non-zero value is considered to be the boolean value TRUE. Here 2 is a non-zero value so TRUE. !TRUE is FALSE (0) so it prints 0.

```

62.  #define FALSE -1
      #define TRUE  1
      #define NULL  0
      main() {
      if(NULL)
          puts("NULL");
      else if(FALSE)
          puts("TRUE");
      else
          puts("FALSE");
      }

```

Answer:
TRUE

Explanation:

The input program to the compiler after processing by the preprocessor is,

```

main(){
    if(0)
        puts("NULL");
    else if(-1)
        puts("TRUE");
    else
        puts("FALSE");
}

```

Preprocessor doesn't replace the values given inside the double quotes. The check by if condition is boolean value false so it goes to else. In second if -1 is boolean value true hence "TRUE" is printed.

Predict the output or error(s) for the following:

```

63.  main()
      {
      int k=1;
      printf("%d==1 is ""%s",k,k==1?"TRUE":"FALSE");
      }

```

Answer:
1==1 is TRUE

Explanation:

When two strings are placed together (or separated by white-space) they are concatenated (this is called as "stringization" operation). So the string is as if it is given as "%d==1 is %s". The conditional operator(?:) evaluates to "TRUE".

```
64.    main()
        {
        int y;
        scanf("%d",&y); // input given is 2000
        if( (y%4==0 && y%100 != 0) || y%100 == 0 )
            printf("%d is a leap year");
        else
            printf("%d is not a leap year");
        }
```

Answer:

2000 is a leap year

Explanation:

An ordinary program to check if leap year or not.

```
65.    #define max 5
        #define int arr1[max]
        main()
        {
        typedef char arr2[max];
        arr1 list={0,1,2,3,4};
        arr2 name="name";
        printf("%d %s",list[0],name);
        }
```

Answer:

Compiler error (in the line arr1 list = {0,1,2,3,4})

Explanation:

arr2 is declared of type array of size 5 of characters. So it can be used to declare the variable name of the type arr2. But it is not the case of arr1. Hence an error.

Rule of Thumb:

#defines are used for textual replacement whereas typedefs are used for declaring new types.

```
66.    int i=10;
        main()
        {
        extern int i;
        {
        int i=20;
        {
        const volatile unsigned i=30;
        printf("%d",i);
        }
        printf("%d",i);
        }
        printf("%d",i);
        }
```

```
}
```

Answer:
30,20,10

Explanation:

'{' introduces new block and thus new scope. In the innermost block i is declared as, `const volatile unsigned` which is a valid declaration. i is assumed of type int. So printf prints 30. In the next block, i has value 20 and so printf prints 20. In the outermost block, i is declared as extern, so no storage space is allocated for it. After compilation is over the linker resolves it to global variable i (since it is the only variable visible there). So it prints i's value as 10.

```
67.  main()
    {
        int *j;
        {
            int i=10;
            j=&i;
        }
        printf("%d",*j);
    }
```

Answer:
10

Explanation:

The variable i is a block level variable and the visibility is inside that block only. But the lifetime of i is lifetime of the function so it lives upto the exit of main function. Since the i is still allocated space, *j prints the value stored in i since j points i.

```
68.  main()
    {
        int i=-1;
        -i;
        printf("i = %d, -i = %d \n",i,-i);
    }
```

Answer:
i = -1, -i = 1

Explanation:

-i is executed and this execution doesn't affect the value of i. In printf first you just print the value of i. After that the value of the expression -i = -(-1) is printed.

```
69.  #include
main()
{
    const int i=4;
    float j;
    j = ++i;
    printf("%d %f", i,++j);
}
```

Answer:
Compiler error

Explanation:
i is a constant. you cannot change the value of constant

```
70. #include  
main()  
{  
  int a[2][2][2] = { {10,2,3,4}, {5,6,7,8} };  
  int *p,*q;  
  p=&a[2][2][2];  
  *q=***a;  
  printf("%d..%d",*p,*q);  
}
```

Answer:
garbagevalue..1

Explanation:
p=&a[2][2][2] you declare only two 2D arrays. but you are trying to access the third 2D(which you are not declared) it will print garbage values. *q=***a starting address of a is assigned integer pointer. now q is pointing to starting address of a.if you print *q meAnswer:it will print first element of 3D array.

```
71. #include  
main()  
{  
  register i=5;  
  char j[]= "hello";  
  printf("%s %d",j,i);  
}
```

Answer:
hello 5

Explanation:
if you declare i as register compiler will treat it as ordinary integer and it will take integer value. i value may be stored either in register or in memory.

```
72. main()  
{  
  int i=5,j=6,z;  
  printf("%d",i+++j);  
}
```

Answer:
11

Explanation:
the expression i+++j is treated as (i++ + j)