

C interview questions and answer with explanation for freshers:

1. Write a c program without using any semicolon which output will : Hello word.

Solution: 1

```
void main(){
    if(printf("Hello world")){
    }
}
```

Solution: 2

```
void main(){
    while(!printf("Hello world")){
    }
}
```

Solution: 3

```
void main(){
    switch(printf("Hello world")){
    }
}
```

2. Swap two variables without using third variable.

```
#include<stdio.h>
int main(){
    int a=5,b=10;
    //process one
    a=b+a;
    b=a-b;
    a=a-b;
    printf("a= %d b= %d",a,b);

    //process two
    a=5;
    b=10;
    a=a+b-(b=a);
    printf("\na= %d b= %d",a,b);
    //process three
    a=5;
    b=10;
    a=a^b;
    b=a^b;
    a=b^a;
    printf("\na= %d b= %d",a,b);
```

```
//process four
a=5;
b=10;
a=b~a-1;
b=a+~b+1;
a=a+~b+1;
printf("\na= %d b= %d",a,b);
```

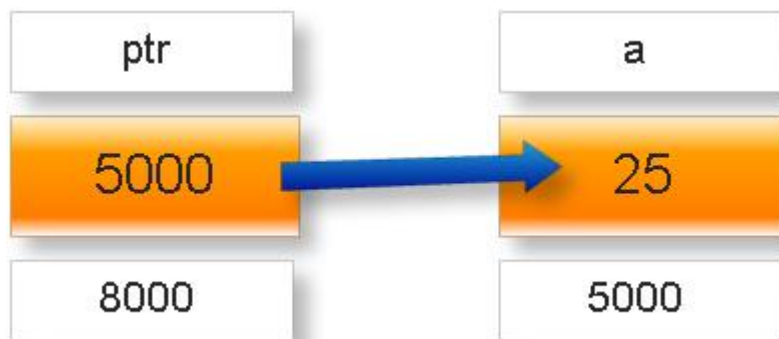
```
//process five
a=5,
b=10;
a=b+a,b=a-b,a=a-b;
printf("\na= %d b= %d",a,b);
getch();
```

```
}
```

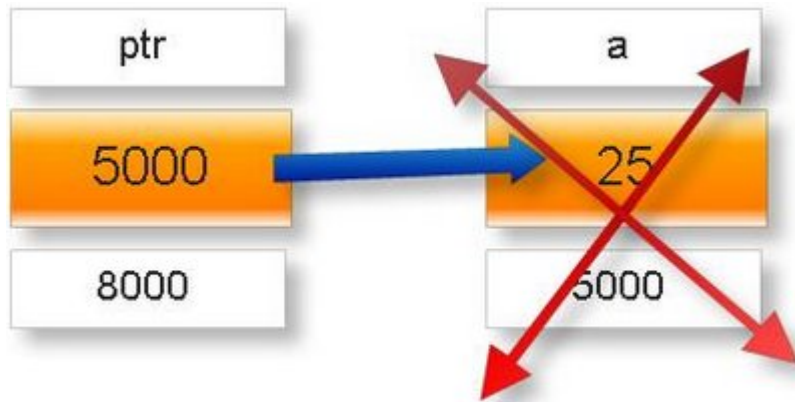
3. What is dangling pointer in c? Answer

If any pointer is pointing the memory address of any variable but after some variable has deleted from that memory location while pointer is still pointing such memory location. Such pointer is known as dangling pointer and this problem is known as dangling pointer problem.

Initially:



Later:



For example:

(q)What will be output of following c program?

```
int *call();
void main(){
    int *ptr;
    ptr=call();
    clrscr();
    printf("%d",*ptr);
}
int * call(){
    int x=25;
    ++x;
    return &x;
}
```

Output: Garbage value

Explanation: variable x is local variable. Its scope and lifetime is within the function call hence after returning address of x variable x became dead and pointer is still pointing ptr is still pointing to that location.

Solution of this problem: Make the variable x is as static variable.

In other word we can say a pointer whose pointing object has been deleted is called dangling pointer.

4. What is wild pointer in c ? Answer

A pointer in c which has not been initialized is known as wild pointer.

Example:

(q)What will be output of following c program?

```
void main(){
    int *ptr;
    printf("%u\n",ptr);
    printf("%d",*ptr);
}
```

Output: Any address
Garbage value

Here ptr is wild pointer because it has not been initialized.

There is difference between the NULL pointer and wild pointer. Null pointer points the base address of segment while wild pointer doesn't point any specific memory location.

Hide

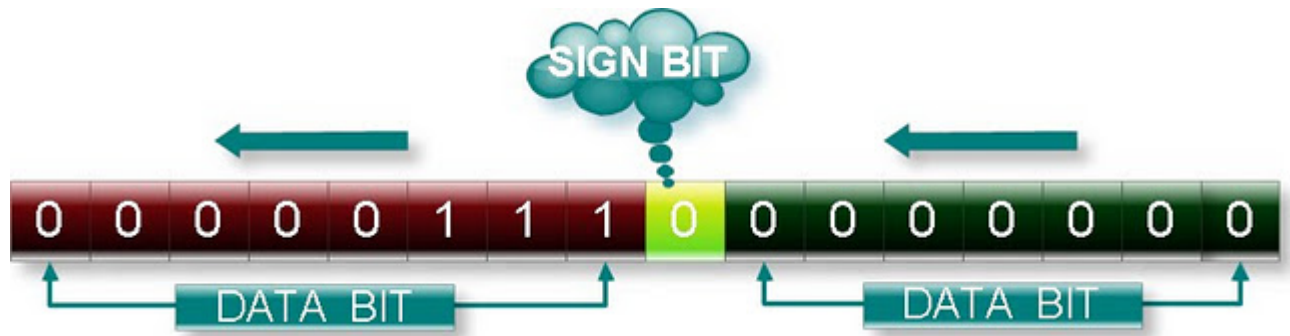
5. What are merits and demerits of array in c? Answer

6. Do you know memory representation of
`int a = 7;` ? Answer

Memory representation of:

signed int a=7; (In Turbo c compiler)
signed short int a=7 (Both turbo c and Linux gcc compiler)
Binary equivalent of data 7 in 16 bit: 00000000 00000111
Data bit: 00000000 00000111 (Take first 15 bit from right side)
Sign bit: 0 (Take leftmost one bit)

First eight bit of data bit from right side i.e. 00000111 will store in the leftmost byte from right to left side and rest seven bit of data bit i.e. 00000000 will store in rightmost byte from right to left side as shown in the following figure:



Hide

7. What is and why array in c ? Answer

An array is derived data type in c programming language which can store similar type of data in continuous memory location. Data may be primitive type (int, char, float, double...), address of union, structure, pointer, function or another array.

Example of array declaration:

```
int arr[5];
char arr[5];
float arr[5];
long double arr[5];
char * arr[5];
int (arr[])();
double ** arr[5];
```

Array is useful when:

(a) We have to store large number of data of similar type. If we have large number of similar kind of variable then it is very difficult to remember name of all variables and write the program.

For example:

```
//PROCESS ONE
```

```
void main(){
    int ax=1;
    int b=2;
    int cg=5;
    int dff=7;
    int am=8;
    int raja=0;
    int rani=11;
```

```

int xxx=5;
int yyy=90;
int p;
int q;
int r;
int avg;
avg=(ax+b+cg+dff+am+raja+rani+xxx+yyy+p+q+r)/12;
printf("%d",avg);
}

```

If we will use array then above program can be written as:

//PROCESS TWO

```

void main(){
    int arr[]={1,2,5,7,8,0,11,5,50};
    int i,avg;
    for(int i=0;i<12;i++){
        avg=avg+arr[i];
    }
    printf("%d",avg/12);
}

```

Question: Write a C program to find out average of 200 integer number using process one and two.

(b) We want to store large number of data in continuous memory location. Array always stores data in continuous memory location.

(q) What will be output when you will execute the following program?

```

void main(){
    int arr[]={0,10,20,30,40};
    char *ptr=arr;
    arr=arr+2;
    printf("%d",*arr);
}

```

Advantage of using array:

1. An array provides single name .So it easy to remember the name of all element of an array.
2. Array name gives base address of an array .So with the help increment operator we can visit one by one all the element of an array.
3. Array has many application data structure.

Array of pointers in c:

Array whose content is address of another variable is known as array pointers. For example:

```

void main(){
    float a=0.0f,b=1.0f,c=2.0f;
    float * arr[]={&a,&b,&c};
    b=a+c;
    printf("%f",arr[1]);
}

```

Hide

8. Why we use do-while loop in c? Also tell any properties which you know ? Answer

It is also called as post tested loop. It is used when it is necessary to execute the loop at least one time.

Syntax:

```

do {
Loop body
} while (Expression);

```

Example:

```

void main(){
    int num,i=0;
    clrscr();
    do{
        printf("To enter press 1\n");
        printf("To exit press 2");
        scanf("%d",&num);
        ++i;
        switch(num){
            case 1:printf("You are welcome\n");break;
            default : exit(0);
        }
    }
    while(i<=10);
    getch();
}

```

Output: 3 3 4 4

If there is only one statement in the loop body then braces is optional. For example:

(a)

```

void main(){

```

```

double i=5.5678;
clrscr();
do
    printf("hi");
while(!i);

    getch();
}

```

Output: 3 3 4 4
(b)

```

void main(){
    double i=5.63333;
    clrscr();
    do
        printf("hi");
    while(!i);

    getch();
}

```

Output: hi

(c)

```

void main(){
    int x=25,y=1;
    do
        if(x>5)
            printf(" ONE");
        else if(x>10)
            printf(" TWO");
        else if(x==25)
            printf(" THREE");
        else
            printf(" FOUR");
        while(y--);
    getch();
}

```

Output: ONE ONE

Hide

9. Why we use static variable in c? Tell me all properties of static variables which you

know? Answer

Keyword static is used for declaring static variables in c. This modifier is used with all data types like int, float, double, array, pointer, structure, function etc. **Important points about static keyword:**

1. It is **not** default storage class of global variables. For example, analyze the following three programs and its output.

(a)

```
#include<stdio.h>
int a;
int main(){
    printf("%d",a);
    return 0;
}
```

Output: 0

(b)

```
#include<stdio.h>
static int a;
int main(){
    printf("%d",a);
    return 0;
}
```

Output: 0

(c)

```
#include<stdio.h>
extern int a;
int main(){
    printf("%d",a);
    return 0;
}
```

Output: Compilation error

At first glance if you will observe the output of above three codes you can say default storage class of global variable is static. But it is not true. Why? Read extern storage class.

2. Default initial value of static integral type variables are zero otherwise null. For example:

```

#include <stdio.h>
static char c;
static int i;
static float f;
static char *str;
int main(){
    printf("%d %d %f %s",c,i,f,str);
    return 0;
}

```

Output: 0 0 0.000000 (null)

3. A same static variable can be declared many times but we can initialize at only one time. For example:

(a)

```

#include <stdio.h>
static int i;    //Declaring the variable i.
static int i=25; //Initializing the variable.
static int i;    //Again declaring the variable i.
int main(){
    static int i; //Again declaring the variable i.
    printf("%d",i);
    return 0;
}

```

Output: 25

(b)

```

#include <stdio.h>
static int i;    //Declaring the variable
static int i=25; //Initializing the variable
int main(){
    printf("%d",i);
    return 0;
}
static int i=20; //Again initializing the variable

```

Output: Compilation error: Multiple initialization variable i.

4. We cannot write any assignment statement globally. For example:

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

```
static int i=10; //Initialization statement
```

```
i=25;           //Assignment statement
```

```
int main(){  
    printf("%d",i);  
    return 0;  
}
```

Output: Compilation error

Note: Assigning any value to the variable at the time of declaration is known as initialization while assigning any value to variable not at the time of declaration is known assignment.

(b)

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

```
static int i=10;
```

```
int main(){  
    i=25;           //Assignment statement  
    printf("%d",i);  
    return 0;  
}
```

Output: 25

(5) A static variable initializes only one time in whole program. For example:

```
#include <stdio.h>
static int i=10;
int main(){
    i=5;
    for(i=0;i<5;i++){
        static int a=10; //This statement will execute
                          //only time.
        printf("%d",a++); //This statement will execute
                          //five times.
    }
    return 0;
}
```

Output: 10 11 12 13 14

(6) If we declared static variable locally then its visibility will be within a block where it has been declared. For example:

```
#include <stdio.h>
int main(){
    {
        static int a=5;
        printf("%d",a);
    }
    //printf("%d",a); variable a is not visible here.
    return 0;
}
```

Output: 5

7. If declared a static variable or function globally then its visibility will be only the file in which it has been declared, not in the other files. For example:

```
(a)
#include <stdio.h>
static float a=144.0f; //global to all function
int main(){
    {
        printf("%d",a); //variable a is visible here.
        //printf("%d",b); variable b is not visible here.
    }
    printf("%d",a); //variable a is visible here.
```

```

    //printf("%d",b);   variable b is not visible here.
    return 0;
}
static int b=5;   //Global to only calculation function
void calculation(){
    printf("%d",a); //variable a is visible here.
    printf("%d",b); //variable b is visible here.
}

```

(b) Consider a c program which has written in two files named as one.c and two.c:

//one.c

```

#include<conio.h>
static int i=25;
static int j=5;

```

```

void main(){
    clrscr();
    sum();
    getch();
}

```

//two.c

```

#include<stdio.h>
extern int i; //Declaration of variable i.
extern int j; //Declaration of variable j.
/**

```

Above two lines will search the initialization statement of variable i and j either in two.c (if initialized variable is static or extern) or one.c (if initialized variable is extern)

*/

```

extern void sum(){
    int s;
    s=i+j;
    printf("%d",s);
}

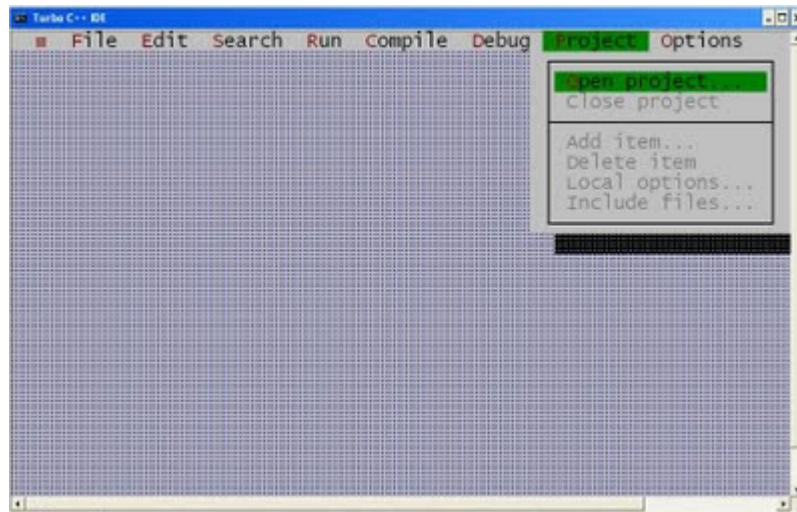
```

Compile and execute above two file one.c and two.c at the same time:

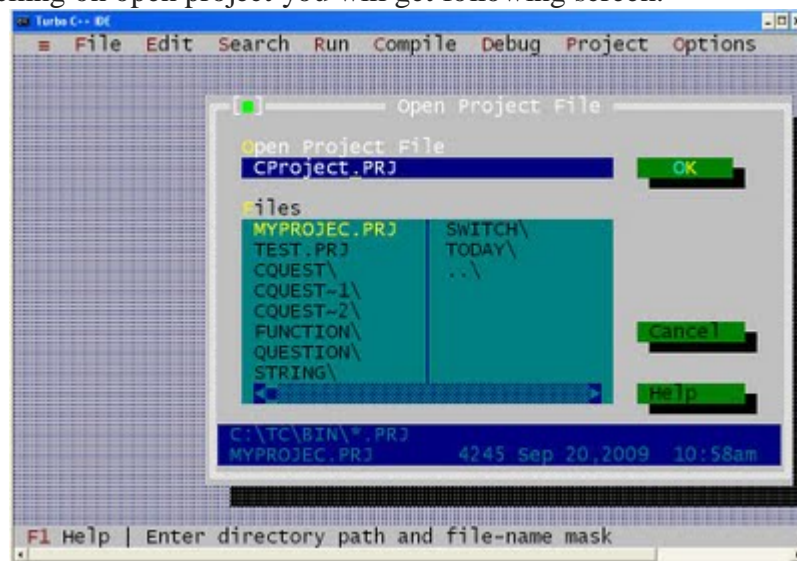
In Turbo c compiler

Step 1: Write above two codes in the file named as one.c and two.c (You can give any name as you like) and save it.

Step 2: In Turbo c++ IDE click on **Project -> Open project** menu as shown in following screen dump.

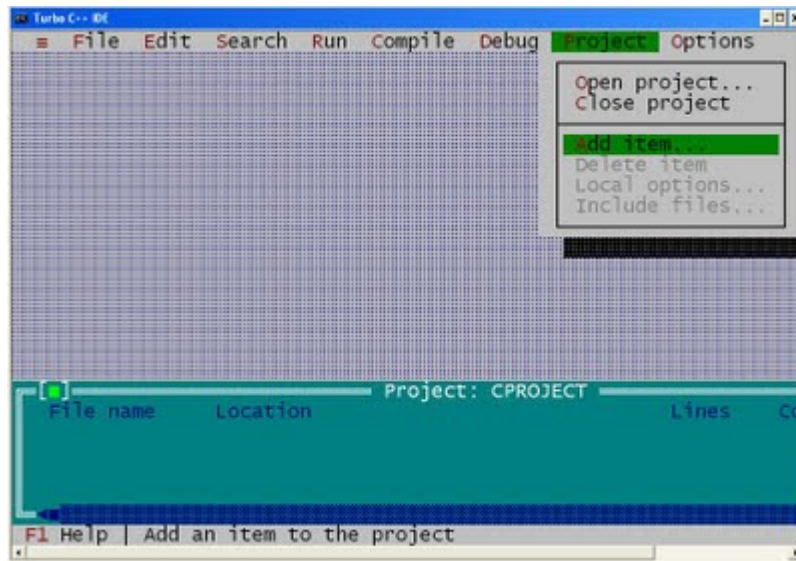


Step 3: After Clicking on open project you will get following screen:



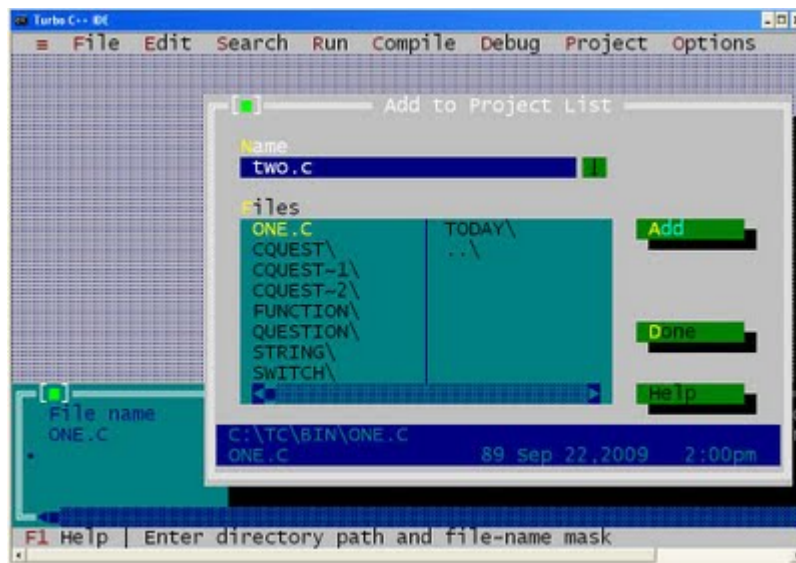
In Open project File text field write any project name with .prj extension. In this example I am writing project name as CProject.PRJ. Now press **OK** button.

Step 4: After pressing OK button you will get following screen:



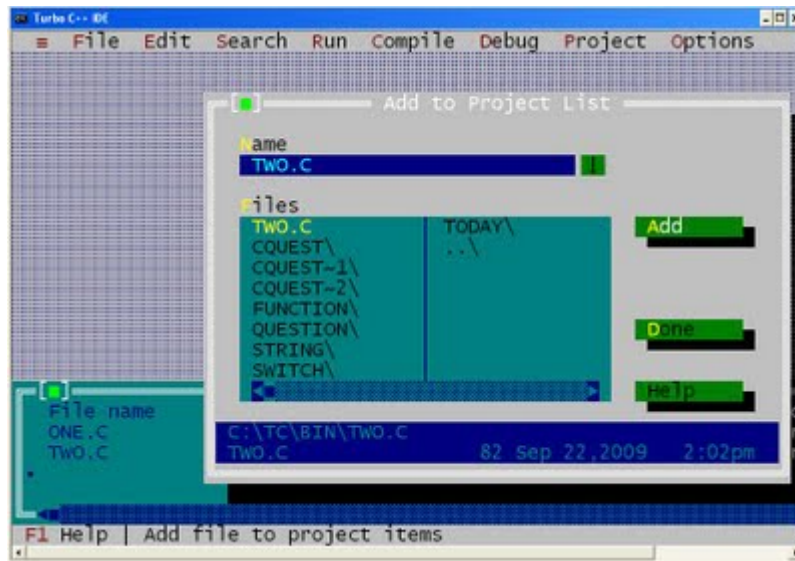
Now click on **Project -> Add item** menu.

Step 5: After clicking Add item you will get following screen:

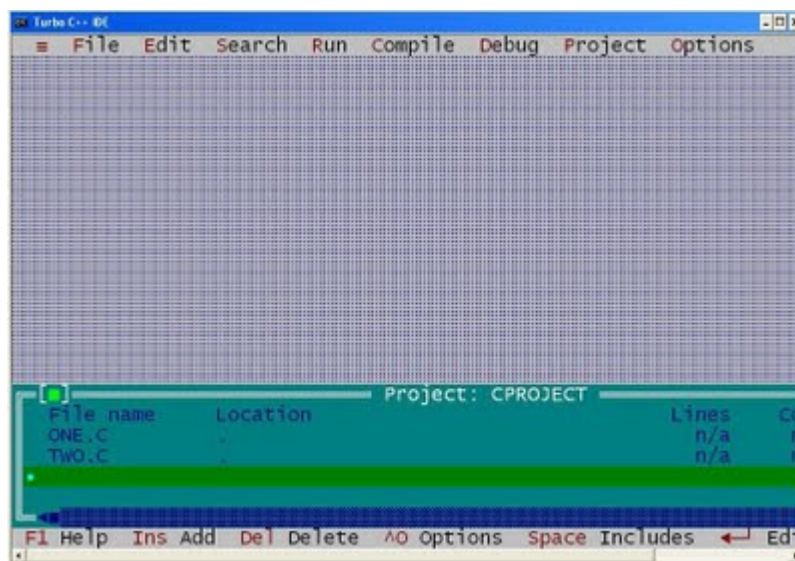


In the name text field write down all c source code file one by one i.e. first write one.c and click on **Add** button

Then write two.c and click on Add button and so on



Step 6: At the end click on **Done** button. After clicking on done button you will get following screen:



At the lower part of window you can see project name, list of files you have added etc.

Step7: To compile the two files press **Alt+F9** and to run the above program press **Ctrl+F9**
 Note: To close the project click on **Project -> Closeproject**.

Output: Compilation error: Unknown symbol i and j.

Hence we can say variable i and j which has initialized into two.c is not visible in file one.c. This example proves visibility of globally declared static variable is file.

Note: In the above example function sum which was declared and defined in two.c has also storage class extern. So we can call from other file (one.c). If it will static then we cannot call function sum since static storage class is only visible to the file where it has declared.

(8) If we static variable has declared locally or globally its scope will always whole the program. For example:

(a) //locally declaration of static variable

```
#include<stdio.h>
void visit();
int main(){
    int i=0;
    {
        //Opening inner block
        static int a=5; //locally declaration
        XYZ;;          //Label of goto statement
        printf("%d ",a);
        a++;
        i++;
    }
    //closing inner block.
    visit();
    /* printf("%d",a); Variable a is not visible here but
    it is alive. */
    if(i<5)
        goto XYZ;
    return 0;
}
void visit(){
}
}
```

Output: 5 6 7 8 9

Explanation: When program control will come out of inner block where variable a has declared then outside of inner block variable a is not visible but its scope is outside the program i.e. variable a hasn't dead. If with help of goto statement control again comes inside the inner block it prints previous incremented values which was not possible in case of auto or register variables.

(b)

//Locally declarations of variable

There are two c source code files:

//one.c

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

```

#include<conio.h>
void main(){
    int i;
    for(i=0;i<3;i++){
        {
            static int a=5;
            printf("%d\n",a);
            a++;
        }
        visit();
    }
    getch();
}
//two.c
#include<stdio.h>
void visit(){
    printf("Don't disturb, I am learning storage class");
    /* printf("%d",a); Variable a is not visible here but
    It is alive. */
}

```

Now compile and execute both files together:

Output:

```

5
disturb, I am learning storage class
6
disturb, I am learning storage class
7
disturb, I am learning storage class

```

Explanation: When control goes to another file and comes even that variable didn't dead and it prints previous incremented value.

Note: In both examples if you will declare static variable globally you will get same output.

9. A static variables or functions have internal linkage. An internal linkage variables or functions are visible to the file where it has declared.

Hide

10. What is the meaning of prototype of a function ? Answer

Prototype of a function

Answer: Declaration of function is known as prototype of a function. Prototype of a function means

- (1) What is return type of function?
- (2) What parameters are we passing?

(3) For example prototype of printf function is:

int printf(const char *, ...);

I.e. its return type is int data type, its first parameter constant character pointer and second parameter is ellipsis i.e. variable number of arguments.

Hide

11. Write a c program to modify the constant variable in c? Answer

You can modify constant variable with the help of pointers. For example:

```
#include<stdio.h>
int main(){
    int i=10;
    int *ptr=&i;
    *ptr=(int *)20;
    printf("%d",i);
}
```

Output: 20

Hide

12. What is the meaning of scope of a variable? Answer

Meaning of scope is to check either variable is alive or dead. Alive means data of a variable has not destroyed from memory. Up to which part or area of the program a variable is alive, that area or part is known as scope of a variable. In the above figure scope of variable is represented outer red box i.e. whole program.

Note: If any variable is not visible it may have scope i.e. it is alive or may not have scope. But if any variable has not scope i.e. it is dead then variable must not be visible.

There are four type of scope in c:

1. Block scope.
2. Function scope.
3. File scope.
3. Program scope.

Hide

13. What is pointer to a function? Answer

(1) What will be output if you will execute following code?

```
int * function();
void main(){
```

```

    auto int *x;
    int (*ptr)();
    ptr=&function;
    x=(*ptr)();
    printf("%d",*x);
}
int *function(){
    static int a=10;
    return &a;
}

```

Output: 10

Explanation: Here function is function whose parameter is void data type and return type is pointer to int data type.

```

x=(*ptr)()
=> x=(*&function)() //ptr=&function
=> x=function() //From rule *&p=p
=> x=&a
So, *x = *&a = a =10

```

(2) What will be output if you will execute following code?

```

int find(char);
int(*function())(char);
void main(){
    int x;
    int(*ptr)(char);
    ptr=function();
    x=(*ptr)('A');
    printf("%d",x);
}
int find(char c){
    return c;
}
int(*function())(char){
    return find;
}

```

Output: 65

Explanation: Here function whose name is function which passing void data type and returning another function whose parameter is char data type and return type is int data type.

```

x=(*ptr)('A')
=> x= (*function ()) ('A') //ptr=function ()
//&find=function () i.e. return type of function ()
=> x= (* &find) ('A')
=> x= find ('A') //From rule *&p=p
=> x= 65

```

(3) What will be output if you will execute following code?

```
char * call(int *,float *);
void main(){
    char *string;
    int a=2;
    float b=2.01;
    char *(*ptr)(int*,float *);
    ptr=&call;
    string=(*ptr)(&a,&b);
    printf("%s",string);
}
char *call(int *i,float *j){
    static char *str="c-pointer.blogspot.com";
    str=str+*i+(int)(*j);
    return str;
}
```

Output: inter.blogspot.com

Explanation: Here call is function whose return type is pointer to character and one parameter is pointer to int data type and second parameter is pointer to float data type and ptr is pointer to such function.

```
str= str+*i+ (int) (*j)
="c-pointer.blogspot.com" + *&a+ (int) (*&b)
//i=&a, j=&b
="c-pointer.blogspot.com" + a+ (int) (b)
="c-pointer.blogspot.com" +2 + (int) (2.0)
="c-pointer.blogspot.com" +4
="inter.blogspot.com"
```

(4) What will be output if you will execute following code?

```
char far * display(char far *);
void main(){
    char far* string="cquestionbank.blogspot.com";
    char far *(*ptr)(char far *);
    ptr=&display;
    string=(*ptr)(string);
    printf("%s",string);
}
char far *display(char far * str){
    char far * temp=str;
    temp=temp+13;
    *temp='\0';
    return str;
}
```

Output: cquestionbak

Explanation: Here display is function whose parameter is pointer to character and return type is also pointer to character and ptr is its pointer.

temp is char pointer

temp=temp+13

temp='\0'

Above two lines replaces first dot character by null character of string of variable string i.e.

"[cquestionbank\0blogspot.com](http://cquestionbank.blogspot.com)"

As we know %s print the character of stream up to null character.

Hide

14. Write a c program to find size of structure without using sizeof operator ? Answer

```
struct ABC{
    int a;
    float b;
    char c;
};
void main(){
    struct ABC *ptr=(struct ABC *)0;
    ptr++;
    printf("Size of structure is: %d",*ptr);
}
```

Hide

15. What is NULL pointer? Answer

Literal meaning of NULL pointer is a pointer which is pointing to nothing. NULL pointer points the base address of segment.

Examples of NULL pointer:

1. **int** *ptr=(**char** *)0;
2. **float** *ptr=(**float** *)0;
3. **char** *ptr=(**char** *)0;
4. **double** *ptr=(**double** *)0;
5. **char** *ptr='\0';
6. **int** *ptr=NULL;

(q) What is meaning of NULL?

Answer:

NULL is macro constant which has been defined in the header file `stdio.h`, `alloc.h`, `mem.h`, `stddef.h` and `stdlib.h` as

#define NULL 0

Examples:

(1)What will be output of following c program?

```
#include "stdio.h"
void main(){
    if(!NULL)
        printf("I know preprocessor");
    else
        printf("I don't know preprocessor");
}
```

Output: I know preprocessor

Explanation:

$!NULL = !0 = 1$

In if condition any non zero number mean true.

(2)What will be output of following c program?

```
#include "stdio.h"
void main(){
    int i;
    static int count;
    for(i=NULL;i<=5;){
        count++;
        i+=2;
    }
    printf("%d",count);
}
```

Output: 3

(3)What will be output of following c program?

```
#include "stdio.h"
void main(){
    #ifndef NULL
    #define NULL 5
    #endif
    printf("%d",NULL+sizeof(NULL));
}
```

Output: 2

Explanation:

$NULL+sizeof(NULL)$

$=0+sizeof(0)$

$=0+2$ //size of int data type is two byte.

We cannot copy any thing in the NULL pointer.

Example:

(q)What will be output of following c program?

```
#include "string.h"
void main(){
    char *str=NULL;
```

```
strcpy(str,"c-pointer.blogspot.com");
printf("%s",str);
}
Output: (null)
```

Hide

16. What is difference between pass by value and pass by reference ? Answer

In c we can pass the parameters in a function in two different ways.

(a)Pass by value: In this approach we pass copy of actual variables in function as a parameter. Hence any modification on parameters inside the function will no reflect in the actual variable.

For example:

```
#include<stdio.h>
void main(){
    int a=5,b=10;
    swap(a,b);
    printf("%d    %d",a,b);
}
void swap(int a,int b){
    int temp;
    temp =a;
    a=b;
    b=temp;
}
```

Output: 5 10

(b)Pass by reference: In this approach we pass memory address actual variables in function as a parameter. Hence any modification on parameters inside the function will reflect in the actual variable. For example:

```
#include<stdio.h>
void main(){
    int a=5,b=10;
    swap(&a,&b);
    printf("%d    %d",a,b);
}
void swap(int *a,int *b){
    int *temp;
    *temp =*a;
    *a=*b;
    *b=*temp;
}
```

Output: 10 5

Hide

17. What is size of void pointer ? Answer

Size of any type of pointer in c is independent of data type which is pointer is pointing i.e. size of all type of pointer (near) in c is two byte either it is char pointer, double pointer, function pointer or null pointer. Void pointer is not exception of this rule and size of void pointer is also two byte.

Hide

18. What is difference between uninitialized pointer and null pointer? Answer

An uninitialized pointer is a pointer which points unknown memory location while null pointer is pointer which points a null value or base address of segment. For example:

```
int *p; //Uninitialized pointer
int *q= (int *)0; //Null pointer
#include<stdio.h>
int *r=NULL; //Null pointer
```

What will be output of following c program?

```
#include<string.h>
#include<stdio.h>
void main(){
    char *p; //Uninitialized pointer
    char *q=NULL; //Null pointer;
    strcpy(p,"cquestionbank");
    strcpy(q,"cquestionbank");
    clrscr();
    printf("%s %s",p,q);
    getch();
}
```

Output: cquestionbank (null)

Hide

19. Can you read complex pointer declaration ? Answer

Rule 1. Assign the priority to the pointer declaration considering precedence and associative according to following table.

| Operator | Precedance | Associative |
|----------------|------------|---------------|
| () , [] | 1 | Left to right |
| * , Identifier | 2 | Right to left |
| Data type | 3 | |

Where

Hide

(): This operator behaves as bracket operator or function operator.

[]: This operator behaves as array subscription operator.

*****: This operator behaves as pointer operator not as multiplication operator.

Identifier: It is not an operator but it is name of pointer variable. You will always find the first priority will be assigned to the name of pointer.

Data type: It is also not an operator. Data types also includes modifier (like signed int, long double etc.)

What is meaning of priority of operator? Click me.

What is meaning of associative of operator? Click me.

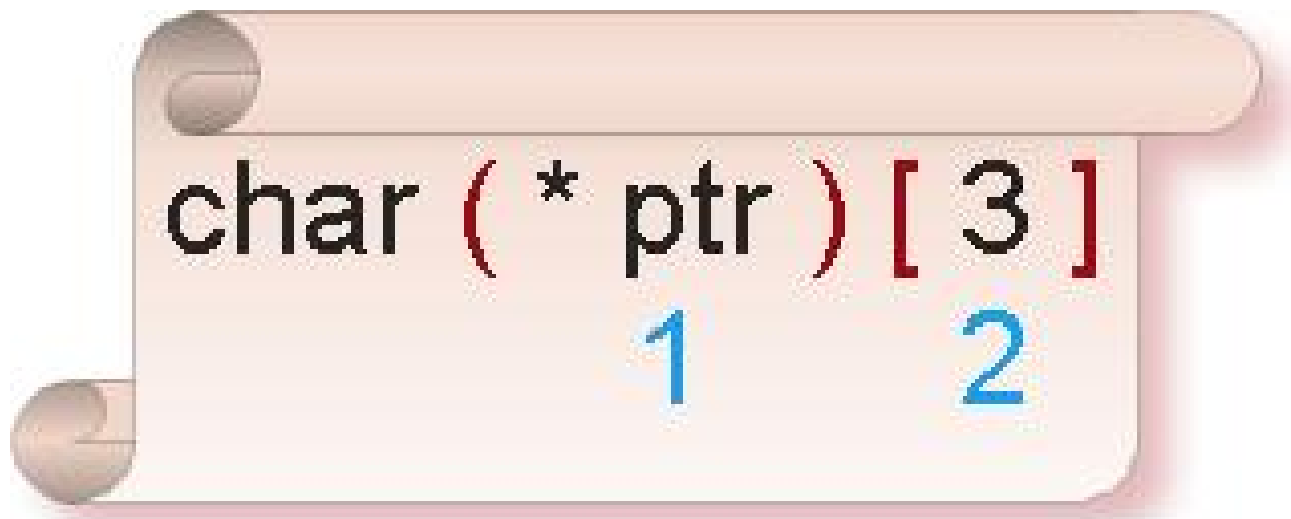
You will understand it better by examples:

(1) How to read following pointer?


`char (* ptr)[3]`

Answer:

Step 1: `()` and `[]` enjoys equal precedence. So rule of associative will decide the priority. Its associative is left to right So first priority goes to `()`.



Step 2: Inside the bracket `*` and `ptr` enjoy equal precedence. From rule of associative (right to left) first priority goes to `ptr` and second priority goes to `*`.



A diagram of a light brown scroll with a darker brown border. The scroll contains the C code `char (* ptr) [3]`. Below the code, the numbers 2 and 1 are written in blue. The asterisk and the variable `ptr` are written in red.

```
char ( * ptr ) [ 3 ]
```

2 1

Step3: Assign third priority to [].



A diagram of a light brown scroll with a darker brown border. The scroll contains the C code `char (* ptr) [3]`. Below the code, the numbers 2, 1, and 3 are written in blue. The asterisk and the variable `ptr` are written in black, and the opening square bracket and the number 3 are written in red.

```
char ( * ptr ) [ 3 ]
```

2 1 3

Step4: Since data type enjoys least priority so assign fourth priority to char.

char (* ptr) [3]

4 2 1 3

Now read it following manner:

ptr is **pointer** to such one dimensional **array** of size three which content **char** type data.

(2) How to read following pointer?

float (* ptr)(int)

Answer:

Assign the priority considering precedence and associative.

float (* ptr) (int)

4 2 1 3

Now read it following manner:

ptr is **pointer** to such **function** whose parameter is int type data and return type is **float** type data.

Rule 2: Assign the priority of each function parameter separately and read it also separately. Understand it through following example.

(3) How to read following pointer?

void (*ptr)(int (*)[2],int (*) void))

Answer:

Assign the priority considering rule of precedence and associative.

```
void ( * ptr ) ( int ( * ) [ 2 ] , int ( * ) ( void ) )
                3 1 2      3 1 2
4 2 1                      3
```

Now read it following manner:

ptr is **pointer** to such **function** which first parameter is **pointer** to one dimensional **array** of size two which content **int** type data and second parameter is **pointer** to such **function** which parameter is void and return type is **int** data type and return type is **void**.

(4) How to read following pointer?

```
int ( * ( * ptr ) [ 5 ] ) ( )
```

Answer:

Assign the priority considering rule of precedence and associative.

```
int ( * ( * ptr ) [ 5 ] ) ( )
6 4 2 1      3 5
```

Now read it following manner:

ptr is **pointer** to such **array** of size five which content are **pointer** to such **function** which parameter is void and return type is **int** type data.

(5) How to read following pointer?

```
double*(*(ptr)(int))(double **,char c)
```

Answer:

```
double * ( * ( * ptr ) ( int ) ) ( double **, char c )
      7   6 4 2 1           3           5
```

Assign the priority considering rule of precedence and associative.

Now read it following manner:

ptr is **pointer** to **function** which parameter is int type data and return type is **pointer** to **function** which first parameter is pointer to pointer of double data type and second parameter is char type data type and return type is **pointer** to **double** data type.

(6) How to read following pointer?

```
unsigned **(*ptr)[8](char const *, ...)
```

Answer:

Assign the priority considering rule of precedence and associative.

```
unsigned * * ( * ( * ptr ) [ 8 ] ( char const *, ... ) )
      8   7 6 4 2 1           3           5
```

Now read it following manner:

ptr is **pointer** to **array** of size eight and content of array is **pointer** to **function** which first parameter is pointer to character constant and second parameter is variable number of arguments and return type is **pointer** to **pointer** of **unsigned** int data type.

20. What are the parameter passing conventions in c ? Answer

1.pascal: In this style function name should (not necessary) in the uppercase .First parameter of function call is passed to the first parameter of function definition and so on.

2.cdecl: In this style function name can be both in the upper case or lower case. First parameter of function call is passed to the last parameter of function definition. It is default parameter passing convention.

Examples:

1.What will be output of following program?

```
void main(){
    static int a=25;
```

```

    void cdecl conv1() ;
    void pascal conv2();
    conv1(a);
    conv2(a);
    getch();
}
void cdecl conv1(int a,int b)
{
    printf("%d %d",a,b);
}
void pascal conv2(int a,int b)
{
    printf("\n%d %d",a,b);
}

```

Output: 25 0
0 25

(2) What will be output of following program?

```

void cdecl fun1(int,int);
void pascal fun2(int,int);
void main(){
    int a=5,b=5;
    clrscr();
    fun1(a,++a);
    fun2(b,++b);
    getch();
}
void cdecl fun1(int p,int q){
    printf("cdecl: %d %d \n",p,q);
}
void pascal fun2(int p,int q){
    printf("pascal: %d %d",p,q);
}

```

Output:
cdecl: 6 6
pascal: 5 6

(3) What will be output of following program?

```

void cdecl fun1(int,int);
void pascal fun2(int,int);
void main(){
    int a=5,b=5;
    clrscr();
}

```



```

    fun1(a,++a);
    fun2(b,++b);
    getch();
}
void cdecl fun1(int p,int q){
    printf("cdecl: %d %d \n",p,q);
}
void pascal fun2(int p,int q){
    printf("pascal: %d %d",p,q);
}

```

Output:

cdecl: 6 6

pascal: 5 6

(4) What will be output of following program?

```

void convention(int,int,int);
void main(){
    int a=5;
    clrscr();
    convention(a,++a,a++);
    getch();
}
void convention(int p,int q,int r){
    printf("%d %d %d",p,q,r);
}

```

Output: 7 7 5

(5) What will be output of following program?

```

void pascal convention(int,int,int);
void main(){
    int a=5;
    clrscr();
    convention(a,++a,a++);
    getch();
}
void pascal convention(int p,int q,int r){
    printf("%d %d %d",p,q,r);
}

```

Output: 5 6 6

(6) What will be output of following program?

```

void pascal convention(int,int);
void main(){
    int a=1;
    clrscr();
    convention(a,++a);
}

```

```

    getch();
}
void pascal convention(int a,int b){
    printf("%d %d",a,b);
}

```

Output: 1 2

(7) What will be output of following program?

```

void convention(int,int);
void main(){
    int a=1;
    clrscr();
    convention(a,++a);
    getch();
}
void convention(int a,int b){
    printf("%d %d",a,b);
}

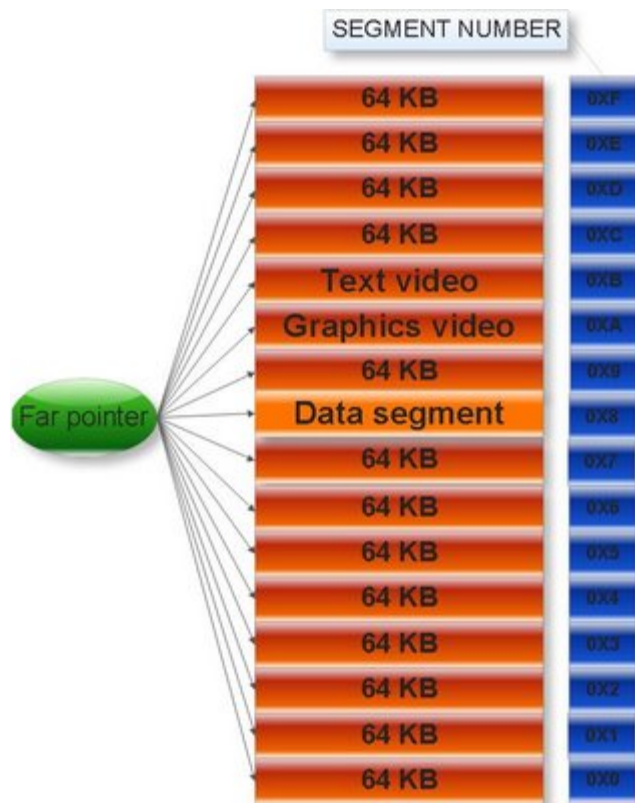
```

Output: 2 2

Hide

21. What is the far pointer in c? Answer

The pointer which can point or access whole the residence memory of RAM i.e. which can access all 16 segments is known as far pointer.



Far pointer:

(If you don't know what is segment the click [here](#))
 Size of far pointer is 4 byte or 32 bit.

Examples:

(1) What will be output of following c program?

```
void main(){
    int x=10;
    int far *ptr;
    ptr=&x;
    printf("%d",sizeof ptr);
}
```

Output: 4

(2) What will be output of following c program?

```
void main(){
    int far *near*ptr;
    printf("%d %d",sizeof(ptr),sizeof(*ptr));
}
```

```
}
```

Output: 4 2

Explanation: ptr is far pointer while *ptr is near pointer.

(3)What will be output of following c program?

```
void main(){
    int far *p,far *q;
    printf("%d %d",sizeof(p) ,sizeof(q));
}
```

Output: 4 4

First 16 bit stores: Segment number

Next 16 bit stores: Offset address

What is segment number and offset address?

Example:

```
void main(){
    int x=100;
    int far *ptr;
    ptr=&x;
    printf("%Fp",ptr);
}
```

Output: 8FD8:FFF4

Here 8FD8 is segment address and FFF4 is offset address in hexadecimal number format.

Note: %Fp is used for print offset and segment address of pointer in printf function in hexadecimal number format.

In the header file dos.h there are three macro functions to get the offset address and segment address from far pointer and vice versa.

1. **FP_OFF():** To get offset address from far address.
2. **FP_SEG():** To get segment address from far address.
3. **MK_FP():** To make far address from segment and offset address.

Examples:

(1)What will be output of following c program?

```
#include "dos.h"
void main(){
    int i=25;
    int far*ptr=&i;
    printf("%X %X",FP_SEG(ptr),FP_OFF(ptr));
}
```

Output: Any segment and offset address in hexadecimal number format respectively.

(2)What will be output of following c program?

```
#include "dos.h"
void main(){
    int i=25;
    int far*ptr=&i;
    unsigned int s,o;
    s=FP_SEG(ptr);
    o=FP_OFF(ptr);
    printf("%Fp",MK_FP(s,o));
}
```

Output: 8FD9:FFF4 (Assume)

Note: We cannot guess what will be offset address, segment address and far address of any far pointer .These address are decided by operating system.

Limitation of far pointer:

We cannot change or modify the segment address of given far address by applying any arithmetic operation on it. That is by using arithmetic operator we cannot jump from one segment to other segment. If you will increment the far address beyond the maximum value of its offset address instead of incrementing segment address it will repeat its offset address in cyclic order.

Example:

(q)What will be output of following c program?

```
void main(){
    int i;
    char far *ptr=(char *)0xB800FFFA;
    for(i=0;i<=10;i++){
        printf("%Fp \n",ptr);
        ptr++;
    }
}
```

```
}
```

Output:

```
B800:FFFA
B800:FFFB
B800:FFFC
B800:FFFD
B800:FFFE
B800:FFFF
B800:0000
B800:0001
B800:0002
B800:0003
B800:0004
```

This property of far pointer is called cyclic nature of far pointer within same segment.

Important points about far pointer:

1. Far pointer compares both offset address and segment address with relational operators.

Examples:

(1)What will be output of following c program?

```
void main(){
    int far *p=(int *)0X70230000;
    int far *q=(int *)0XB0210000;
    if(p==q)
        printf("Both pointers are equal");
    else
        printf("Both pointers are not equal");
}
```

Output: Both pointers are not equal

(2)What will be output of following c program?

```
void main(){
    int far *p=(int *)0X70230000;
    int far *q=(int *)0XB0210000;
    int near *x,near*y;
    x=(int near *)p;
    y=(int near *)q;
```

```

    if(x==y)
        printf("Both pointer are equal");
    else
        printf("Both pointer are not equal");
}

```

Output: Both pointers are equal

2. Far pointer doesn't normalize.

Hide

22. Do you know pointer to function? Explain it by any example? Answer

A pointer which holds address of a function is known as pointer to function. Example:

```

int * function();
int *(*ptr)();
ptr=&function;

```

Here ptr is pointer to function.
Complete program:

```

int * function();
void main(){
    auto int *x;
    int *(*ptr)();
    ptr=&function;
    x=(*ptr)();
    printf("%d",*x);
}
int *function(){
    static int a=10;
    return &a;
}

```

Output: 10

Explanation: Here function is function whose parameter is void data type and return type is pointer to int data type.

```
x>(*ptr)()
=> x=(*&function)() //ptr=&function
=> x=function() //From rule *&p=p
=> x=&a
So, *x = *&a = a = 10
```

Hide

23. What is meaning of continue keyword in c ? Answer

It is keyword of c and task of this keyword is to transfer the control of program at the beginning of loop. For example:

```
(a)
void main(){
    int i=5;
    clrscr();
    do{
        printf("%d",i);
        continue;
        i++;
    }
    while(i<=10);
    getch();
}
```

Output: Infinite loop

```
(b)
void main(){
    int i=5;
    clrscr();
    do{
        printf("%d",i);
        continue;
        i++;
    }
    while(i<=10);
    getch();
}
```


Output: Infinite loop

Except looping ,we cannot use continue keyword.

```
void main(){
    int x;
    scanf("%d",&x);
    clrscr();
    switch(x){
        case 1:printf("1");break;
        case 2:printf("2");continue;
        default:printf("3");
    }
    getch();
}
```

Output: Compilation error

Hide

24. Tell me all the properties of variables in c? Answer

Every variable in c have three most fundamental attributes. They are:

1. Name
2. Value
3. Address

Name of a variable:

Every variable in c has its own name. A variable without any name is not possible in c. Most important properties of variables name are its unique names. Not two variables in c can not have same name with same visibility. For example:

(a)

```
#include<stdio.h>
#include<conio.h>
void main(){
    int a=5;
    int a=10;
    /* Two variables of same name */
    printf("%d",a);
    getch();
}
```

(b)

```
#include<stdio.h>
#include<conio.h>
void main(){
    auto int a=5; //Visibility is within main block
    static int a=10; //Visibility is within main block
```

```

/* Two variables of same name */
printf("%d",a);
getch();
}

```

Output: compilation error

Output: Compilation error

But it is possible that two variable with same name but different visibility. In this case variable name can access only that variable which is more local. In c there is not any way to access global variable if any local variable is present of same name. For example:

(a)

```

#include<stdio.h>
#include<conio.h>
int a=50; //Visibility is whole the program
void main(){
    int a=10; //Visibility within main block
    printf("%d",a);
    getch();
}

```

Output: 10

(b)

```

#include<stdio.h>
#include<conio.h>
void main(){
    int a=10; //Visibility within main block.
    {
        a+=5; //Accessing outer local variable a.
        int a=20; //Visibility within inner block.
        a+=10; //Accessing inner local variable a.
        printf("%d",a); //Accessing inner local variable a.
    }
    printf("%d",a); //Accessing outer local variable a.

    getch();
}

```

Output: 30 15

Note: In c any name is called identifier. This name can be variable name, function name, enum constant name, micro constant name, goto label name, any other data type name like structure, union, enum names or typedef name.

Value of variable:

Data which any variable keeps is known as value of variable. For example:

```
int a=5;
```

Here value of variable a is five. Name of variable always returns value of the variable.

How to assign any value to a variable:

C supports eleven type of assignment operator to assign any value to operator. Those are:

(a) = (b) += (c) -= (d) *= (e) /= (f) %=

(g) <<= (h) >>= (i) |= (j) &= (k) ^=

In this chapter will discuss only first operator i.e. =

Assignment statement in c:

LValue: Lvalue stands for left value. In any assignment statement LValue must be a container i.e. which have ability to hold the data. In c has only one type only container and which is variable. Hence LValue must be any variable in c it cannot be a constant, function or any other data type of c. For example

```
#define <stdio.h>
#define <conio.h>
#define max 125
struct abc{
    char *name;
    int roll;
};
enum {RED,BLUE};
void main(){
    int a;
    const b;
    a=5;
    //10=5; LValue cannot be a integer constant
    //max=20; //Lvalue cannot be a micro constant
    //b=11; Lvalue cannot be a constant variable
    //float=3.3f; Lvalue cannot be a data type
    //abc={"sachin",5}; Lvalue cannot be a data type
    //BLUE =2; Lvalue cannot be a enum constant
    getch();
}
```

RValue: In any assignment statement RValue must be any thing which can return and constant value or it is itself a constant. RValue can be any c constants, variables, function which is returning a value etc. For example:

```
#include<stdio.h>
#include<conio.h>
#define max 5
```

```

void display();
float sum();
enum {BLACK, WHITE};
void main(){
    int x=2; //RValue can be a constant.
    float y=9.0;
    const int z=x; //RValue can be a constant variable
    x=max; //RValue can be a variable.

    x=BLACK; //RValue can be a enum constant.
    y=sum(); //RValue can be a function.
    y=display(); //RValue can be a function which is
                //returning nothing.
}

```

Hide

25. What is void data type? Tell me any three use of void data type. Answer

Linguistic meaning of void is nothing. Size of void data type is meaningless question.

What will be output of following c code?

```

#include<stdio.h>
int main(){
    int size;
    size=sizeof(void);
    printf("%d",size);
    return 0;
}

```

Output: Compilation error

If we will try to find the size of void data type compiler will show an error message “not type allowed”. We cannot use any storage class modifier with void data type so void data type doesn’t reserve any memory space. Hence we cannot declare any variable as of void type i.e.

```
void num; //compilation error
```

Use of void data type

1. To declare generic pointer
2. As a function return type
3. As a function parameter.

Hide

26. What is constant variable in c? What are the benefits of using constant variable? Answer

In c all variables are by default not constant. Hence, you can modify the value of variable by program. You can convert any variable as a constant variable by using modifier const which is keyword of c language.

Properties of constant variable:

1. You can assign the value to the constant variables only at the time of declaration. For example:

```
const int i=10;
float const f=0.0f;
unsigned const long double ld=3.14L;
```

2. Uninitialized constant variable is not cause of any compilation error. But you can not assign any value after the declaration. For example:

```
const int i;
```

If you have declared the uninitialized variable globally then default initial value will be zero in case of integral data type and null in case of non-integral data type. If you have declared the uninitialized const variable locally then default initial value will be garbage.

3. Constant variables executes faster than not constant variables.

Hide

27. Do you know any post tested loop in c ? Answer

While loop:

It is pre tested loop. It is used when we have to execute a part of code in unknown numbers of times.

Syntax:

```
while (Expression){
Loop body
}
```

Properties of while loop:

1. Task of the expression is to check the condition. Loop will execute until condition is true otherwise loop will terminate.

2. If any expression returns zero then condition will false and if it returns any non- zero number then condition will true. For example:

(a)

```
void main(){
    int x=3,y=2;
    clrscr();
    while(x+y-1){
        printf("%d ",x--+y);
    }
    getch();
}
```

Output: 5 4 3 2

(b)

```
void main(){
    float a=1.5f;
    clrscr();
    while(a){
        printf("%.f ",a);
        a-=.5f;
    }
    getch();
}
```

Output: 2 1 0

3. In while loop condition expression is compulsory. For example:

```
void main(){
    clrscr();
    while(){
        printf("Hello world");
    }
    getch();
}
```

Output: Compilation error

4. while loop without any body is possible. For example:

```
void main(){
    int i=0;
    clrscr();
    while(i++,i<=8);
    printf("%d ",i);
    getch();
}
```

```
}
```

Output: 9

5. In while loop there can be more than one conditional expression. For example

```
void main(){
    int x=2,y=2;
    clrscr();
    while(x<=5,y<=3)
        printf("%d %d ",++x, ++y);
    getch();
}
```

Output: 3 3 4 4

6. If loop body contain only one statement the brace is optional. For example:

```
void main(){
    clrscr();
    while(!printf("Hello world"));
    getch();
}
```

Output: Hello world

Hide

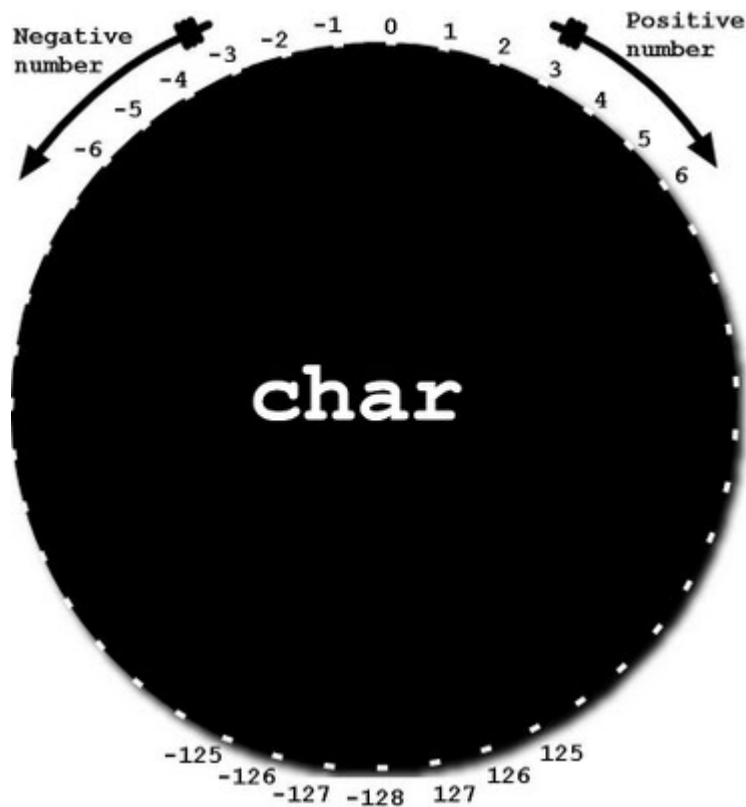
28. What is cyclic properties of data type in c? Explain with any example. Answer

```
#include<stdio.h>
int main(){
    signed char c1=130;
    signed char c2=-130;
    printf("%d %d",c1,c2);
    return 0;
}
```

Output: -126 126 (why?)

This situation is known as overflow of signed char.

Range of unsigned char is -128 to 127. If we will assign a value greater than 127 then value of variable will be changed to a value if we will move clockwise direction as shown in the figure according to number. If we will assign a number which is less than -128 then we have to move in anti-clockwise direction.



Hide

29. Do you know nested loop. Explain by an example ? Answer

A loop inside another loop is known as nested loop. We can write any loop inside any loop in c i.e. we can write for loop inside the loop or while loop or do while loop etc. For example:

(a)

```
void main(){
    int i,j,k;
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            printf(" %d",i+j);
        }
    }
    getch();
}
```


(b)

```
void main(){  
    int i,j,k;  
    do  
    while(0)  
        for(;0;)  
            printf("cbyexample");  
    while(0);  
    getch();  
}
```

Hide

30. Could you define variable in c? Answer