

How to get success in CPU?

- 1) Do you know importance of C language?
- 2) Have you prepared the fundamental aspects of C language?
- 3) Are you aware with prerequisites of C language?
- 4) Have you tried to execute simple C block of code by your own self?
- 5) Do you know that C is equally important for other discipline as Computer Engineering and Information Technology?
- 6) Do you know that C is more useful for the higher semester curriculum directly or indirectly?
- 7) Do you know you can develop many graphical and animated projects using C language?

To know the answer of above questions or to be able to give the answer of those, there are some guidelines given in this manual which could help you to get started and get success on the track of C and its applications. The guidelines may encourage and empower you to have the strong grip on C. So just concentrate on what is your goal? Want to learn this language thoroughly? Then just go through this manual. I am sure it will definitely help you out.

You must be clear about the following concepts in order to get success in CPU.

- 1) 'C' Development life cycle using flowchart in detail.
- 2) Components of computer.
- 3) Flowcharts like
 - a. Find the smallest number out of three given numbers.
 - b. Count odd numbers and making their total from given N numbers.
- 4) Basic features of 'C' language.
- 5) Data types in C.
- 6) Basic difference between
 - a. Procedure Oriented and Object Oriented Programming?
 - b. Switch case and Nested If
 - c. Break and Continue
 - d. Structure and Union
 - e. While and Do-While
 - f. malloc and calloc
 - g. getchar() and putchar()
 - h. gets() and puts()
 - i. call by value and call by reference
- 7) Types of functions.
- 8) Operator and their precedence in C.
- 9) String manipulation functions: strcpy(), strcat(), strcmp(), strlen().
- 10) One dimensional array and two dimensional array.

How to attempt paper of CPU?

- It is better to write paper point-wise rather than write in paragraph.

Let say there is a question like:

Que: what are the differences between Procedures oriented and Object Oriented Programming?

Procedure Oriented	Object Oriented
1. Main program is divided into small parts depending on the functions.	Main program is divided into small object depending on the problem.
2. Every function contains different data.	Data & functions of each individual object act like a single unit.
3. Functions get more importance than data in program	Data gets more importance than functions in program.
4. Most of the functions use global data.	Each object controls its own data.
5. Same data may be transfer from one function to another	Data does not possible transfer from one object to another.
6. There is no perfect way for data hiding.	Data hiding possible in OOP which prevent illegal access of function from outside of it. This is one of the best advantages of OOP also.
7. To add new data in program user should be ensure that function allows it.	Message passing ensure the permission of accessing member of an object from other object.
8. Top down process is followed for program design.	Bottom up process is followed for program design.
9. Eg. C , Pascal, Fortran	Eg. C++, Java

Que. What are the difference between Structure and Union?

Structure	Union
1. It allocates memory equal to sum of memory allocated to its each individual member.	1. It allocates piece of memory that is Large enough to hold the Largest variable of type in union.
2. Each member have their own memory space	2. One block is used by all the members of union.
3. Structure cannot be implemented in shared memory.	3. Union is the Best environment where memory is shared
4. It has less Ambiguity.	4. As memory is shared, Ambiguity is more in union.
5. Self referential structure can be implemented in data structure.	5. Self ref. union cannot be implemented.
6. All members if structure can be accessed at a time.	6. Only one member is accessed at a time.

➤ The programs should be in structured format.

for example

Que: Write a program to add digits of number using C programming language.

```
#include<stdio.h>
#include<conio.h>
main()
{
int n, sum = 0, remainder;
printf("Enter an integer\n");
scanf("%d",&n);
while( n != 0 )
{ remainder = n % 10;
sum = sum + remainder;
n = n / 10;
}
printf("Sum of digits of entered number = %d\n",sum);
return 0;
}
```

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

main()
{
int n, sum = 0, remainder;

printf("Enter an integer\n");
scanf("%d",&n);

while( n != 0 )
{
remainder = n % 10;
sum = sum + remainder;
n = n / 10;
}

printf("Sum of digits of entered number = %d\n",sum);

return 0;
}
```

Never write a program on single alignment.

- Give the example whenever you are asked to explain any concept like looping statement, decision making statements, array etc.

For example

Que: Explain if Statement:-

The **if** statement is the simplest decision making statement. It is used to decide whether to do something at a special point, or to decide between two courses of action.

Syntax:-

<u>Single statement body</u>	<u>Multiple statement body</u>
<pre>if (expression) Statement 1; else Statement 2;</pre>	<pre>if (expression) { Statement 1 ; Statement 1; . . } else { Statement2 ; Statement 2; . . }</pre>

Here the else part is optional. The test expression within the brackets following the **if** is evaluated; if it is true (i.e. expression has a non zero value) then statement1 is executed. If it is false(i.e expression has zero value) and if there is **else** part, statement 2 is executed instead.

Example:-

Single statement if body

```
if (marks > 35)
    printf("Pass");
else
    printf("Fail");
```

Multiple statement if body

```
if (marks > 35)
{
    Printf("Pass");
    Printf("Congratulations");
}
else
{
    Printf("Fail");
    Printf("Better luck next time");
}
```

Que : Explain for loop:-

Ans: The for loop is most commonly used loop statement. It is pre-test loop and it is used when the number of iterations of the loop is known before the loop is entered.

Syntax:-

```
for (initialization expression ;test expression ; update expression)
```

```
    statement;
```

or

```
for (initialization expression ;test expression ; update expression)
```

```
{
```

```
    statement;
```

```
    statement;
```

```
    statement;
```

```
}
```

Here the initialization expression is run before the loop is entered. Loop variable is initialized here only once.

Next the test expression - it is checked at every entry of the loop. The loop is terminated when the test expression returns false else loop is continued.

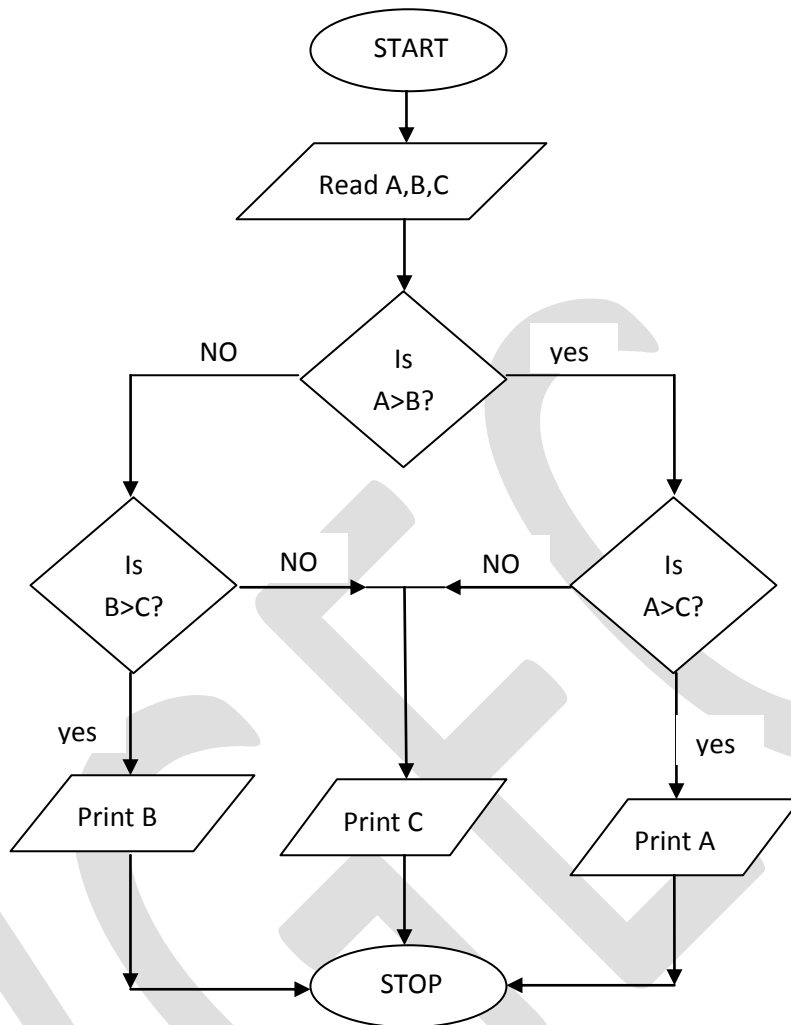
Third part is run every time the loop body is completed. This is usually an increment or decrement of the loop counter.

Example:-

```
int main()
{
    int i;
    for (i = 0; i < 10; i++)
    {
        printf ("Hello\n");
        printf ("World\n");
    }
    return 0;
}
```

In the example i is loop variable and it is initialized to the value of 0 [i = 0]. we used i++ which is the same as using i = i + 1. This is called incrementing. The instruction i++ adds 1 to i. If you want to subtract 1 from i you can use i-. It is also possible to use ++i or --i. The difference is that with ++i the one is added before the “for loop” tests if i < 10. With i++ the one is added after the test i < 10. What so ever be the update statement, the expression “i < 10” is checked all time at the entry of the loop. The block is executed 10 times i.e until the value of i is less than 10. When it becomes 11 the loop is exited.

Que : Draw the flow chart for finding largest number from the three given number.



Hope this guideline will help you get success in CPU.

----- ALL THE BEST -----