Looping Commands

If a set of instructions are to be executed again and again, we resort to looping commands. 3 different commands are available.

1. while
2. for
3. do……while

The 1st 2 types are also known as “top tested loops” where as the last is called as “bottom tested loop”. Top tested loop means the condition in front of “while” or “for” is 1st checked & if true, then the loop executes. However in bottom tested loop, there is always a guarantee that the loop will execute atleast once.

Syntax of “while” command

if condition turns false, loop terminates

while (condition/s) recheck the condition

{ true

do this1;

do this2;

do this3;

……..

}

do this;

You may observe similarity between the “if” command & “while” command except that “else” is not available for “while”. Also after execution of code from scope of “while” command { }, the control of the program is again sent to “while” command to “recheck the condition”. Suppose the condition is satisfied/true, then the code in scope of “while” command will be executed again. This goes on & on. Now the statements inside “while” loop should be such that at some point of time, the condition in front of “while” should turn false, so that the loop terminates. If this doesn’t happen, then the loop executes/iterates “infinite times” which is treated as an error called as “infinite loop” by programmers.

Keyboard

12 ENTER

144

12

no sq ask

buffer

RAM

ASCII Value of ENTER & char ‘y’ is different

ENTER

Data from keyboard is 1st sent to physical memory in RAM called as “Buffer”. Initially when the scientists transferred the data directly to a variable, they observed that when next digit/char is typed, it overwrites the previous ones. So if we type 1956, it was observed that var contained last digit i.e. 6 only. The buffer will flush the data to var only after we press RETURN/ENTER key.

Syntax of “for” loop

for (initial value of var;condition on var;increment/decrement value of var)

{ true

do this1;

do this2;

do this3;

………

}

do this;

Execution steps are 1-2- 3-4-5-3-4-5-3-4-5…….. till 5 becomes false

while for

|  |  |
| --- | --- |
| int count = 0;  …..  while (count < 3)  {  ….  ….  count++; // same as ++count  } | int count; // only declaration  for (count=0;count<3;count++)  {  …..  …..  } |
| char ask = ‘y’;  ……  while (ask == ‘y’ || ask == ‘Y’)  {  …….  printf(“Continue ?”);  scanf(“%c”,&ask);  } | char ask;  …….  for (ask=’y’; ask == ‘y’ || ask == ‘Y’; ) // 2 ;s compuslory  {  …………  printf(“Continue ?”);  scanf(“%c”,&ask);  } |
| char ask;  ….  while (1) // self initialised loop  {  ….  If (condition)  break; // compulsory  } | char ask;  ….  for (; ;) // self initialised loop  {  ….  If (condition)  break; // compulsory  } |
| while loop is preferred if loop iterations are controlled by char var | for loop is preferred if loop iterations are controlled by numeric var. In char vars, as there is no increment/decrement of var value. So for loop is not preferred |

Syntax of do….while loop

do

{

do this1;

do this2;

………

……..

} while (condition);

The loop executes atleast once