**Repetition statements in C**

**Counter-Controlled Repetition**

* Number of repetitions is known before the loop begins execution.
* A control variable is used to count the number of repetitions.
* The control variable is incremented (usually by 1) each time the group of instructions is performed.
* The repetition terminates when the counter exceeds number of repetitions.

**Counter Controlled Repetition cont…**

* Counter-controlled repetition requires:
  + The name of a control variable.
  + The initial value of the control variable
  + The increment ( or decrement) by which the control variable is modified each time through the loop.
  + The condition that tests for the final value of the control variable.

**Counter-Controlled Repetition with the while statement**

//Counter-controlled repetition

# include <stdio.h> int main(void)

{

int counter = 1; // initialization

while (counter <= 10) { //repetition condition printf(“%d ”, counter); // display counter

++ counter; // increment

} // end while

} //end function main

Output

1 2 3 4 5 6 7 8 9 10

**General Format of a while Statement**

while (condition) {

statements

}

* The while statement body may contain single or a compound statement.

**Sentinel-Controlled Repetition**

* When no indication is given of how many times the loop should execute, a

sentinel value is used to terminate the loop.

* E.g : type -1 to terminate entering of marks
* A loop should have a statement to obtain data each time the loop is performed.
* sentinel value must chosen so that it cannot be confused with an acceptable input value.

// class average program with sentinel-controlled repetition # include <stdio.h>

int main(void)

{

int grade, total, counter;

float average;

total = 0;

counter = 0;

//get first input from the user

printf(“Enter grade, -1 to end :”);

scanf(“%d”,&grade);

while(grade != -1){

total = total + grade;

counter = counter + 1;

// get next grade from user printf(“Enter grade, -1 to end :”); scanf(“%d”,&grade);

} //end while

average = (float)total / counter;

printf(“Class average is %.2f\n”, average);

}//end function main

**Output**

Enter grade, -1 to end : 75

Enter grade, -1 to end : 94

Enter grade, -1 to end : 97

Enter grade, -1 to end : 88

Enter grade, -1 to end : 70

Enter grade, -1 to end : 64

Enter grade, -1 to end : 83

Enter grade, -1 to end : 89

Enter grade, -1 to end : -1

Class average is 82.50

**Counter-Controlled Repetition with the for statement**

# include <stdio.h>

int main(void)

{

int counter; // define counter

for( counter = 1; counter <= 10; ++counter ){

printf(“%d\n”, counter);

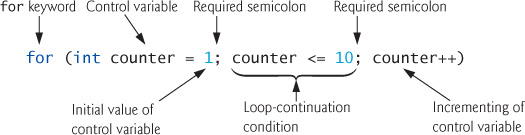
}

}

**Output**

1 2 3 4 5 6 7 8 9 10

**for Statement Header Components**



**General Format of a for Statement**

for ( expression1; expression2; expression3){

statement

{

* Expressions in the for statement’s header are optional
* Increment Expression acts like a standalone statement

**do…while Repetition Statement**

* Loop continuation condition is checked after the loop body is performed.
* Therefore the loop body will be executed at least once.

do{

statement

}while (condition);

**Counter-Controlled Repetition with the do…while statement**

# include <stdio.h>

int main(void)

{

int counter = 1;

do{

printf(“%d ”, counter);

} while (++counter <= 10);

}

**Output**

1 2 3 4 5 6 7 8 9 10

**break statement**

* The break statement, when executed in a while, for, do…while statement causes immediate exit from that statement.
* Program execution continues with the next statement.
* Common uses of the break statement are to escape early from a loop.

**break statement example**

//Using a break statement in a while statement

# include <stdio.h> int main(void)

{

int x = 1;

while ( x <= 10) {

if ( x == 5) { break;

}

printf(“%d ”, x);

++ x;

} // end while

} //end function main

**Output**

1 2 3 4

**continue statement**

* The continue statement, when executed in a while, for and do...while statement, skips the remaining statements in the body of that control statement and perform the next iteration of the loop.
* In while and do...while , loop continuation test is evaluated immediately after the continue statement is executed.
* In the for statement, the increment expression is executed.

**continue statement example**

//Using the continue statement in a while statement

# include <stdio.h> int main(void)

{

int x = 1;

while ( x <= 10) {

if ( x == 5) {

++x;

continue;

}

printf(“%d ”, x);

++ x;

} // end while

} //end function main

**Output**

1 2 3 4 6 7 8 9 10

**Nested iteration**

# include <stdio.h>

int main(void)

{

int i, j;

for ( i = ; i <= 5 ; ++i){

for ( j = 1; j <= i; ++j){

printf(“ \*” );

}

printf(“\n”);

}

}

**Output**

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*

\*