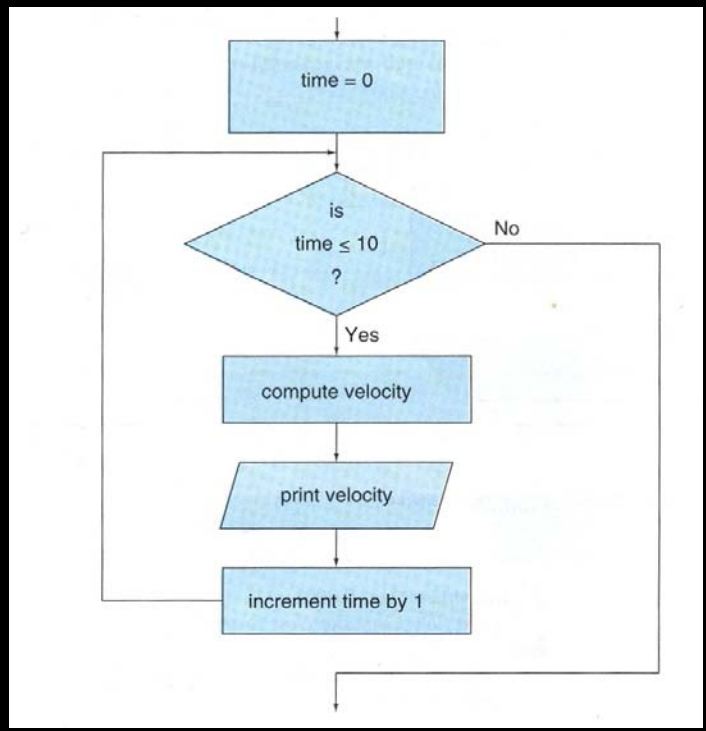
BIL 104E Introduction to Scientific and Engineering Computing

Lecture 5

Loop Structures

Loop Structures

- Loops are used to implement repetitive structures. C contains three repetitive structures:
  - **while** loop
  - **do/while** loop
  - **for** loop




```
set time to 0
while time≤10
        compute velocity
        print velocity
        increment time by 1
```

while Loop

- **General Form:**

```
initialization;
while (loop continuation test){
        statement(s);
        increment;
    }
```

- **Example:**

```
i = 1;
while ( i<=10){
    printf("%d ", i);
    i++;
}
```

OR

```
i = 1;
while ( i<=10)
    printf("%d ", i++);
```

- The condition is evaluated **before** the statement within the loop is executed.
- If the condition is **false** then the loop is skipped,
- else if the condition is **true** then the loop statement is executed and the condition is evaluated again.
- If it is still **true** the statement is executed again and the condition is evaluated again.
- This repetition continues until the condition becomes **false**.

while Loop

- The statement within the loop must modify variables that are used in the condition; otherwise, the value of the condition will never change, and we will never be able to exit the loop. If the condition in a while loop is always true then an **infinite loop** is generated.

```
#include <stdio.h>
int main(){
    int a, b,c;
    while(1){
        printf("Enter three integers ? : ");
        scanf("%d%d%d",&a,&b,&c);
        printf("Given a = %d, b = %d, c = % d \n",a,b,c);
        if ( a+b > c) printf(" a+b > c \n");
        if ( a+b < c) printf(" a+b < c \n");
        if ( a+b == c) printf(" a+b = c \n");
     }
    return 0;
}
```

do/while Loop

- **General Form:**

```
do{
      statement(s);
}while (condition);
```

- **Example:**

```
counter=1;
do {
   printf("%d ", counter);
}while (++counter <= 10);
```

OR

```
counter=1;
do {
   printf("%d ", counter++);
}while (counter <= 10);
```

- The condition is tested at the end of the loop. This ensures that the loop is always executed at least once.

Output: 1 2 3 4 5 6 7 8 9 10

for Loop

- **General Form:**

```
for (initialization; loop continuation test; increment)
{
        statement(s);
}
```

- "for loop" is based on the value of a variable that increments (or decrements) by the same amount each time through the loop. When the variable reaches a specified value the loop is exited.

- **Example:**

```
/*Print the integers from one to ten*/
for (counter = 0; counter <= 10; counter++)
        printf("%d\n", counter);
```

Output: 0 1 2 3 4 5 6 7 8 9 10

for Loop

```
/*Series summation*/
sum=0;
for (number=2; number <= 100; number +=2)
            sum += number;
printf("Sum is %d\n", sum);
```

```
/*Can be comma-separated lists*/
for (i = 0, j=0; j+i <= 10; j++, i++)
            printf("%d\n", j + i);
```

```
/*arithmetic expressions can be used */
for ( j = x; j <= 4 * x * y; j += y / x )
```

```c
#include <stdio.h>

int square( int y );   /* function prototype */

int main()
{
     int x; /* counter */

     for ( x = 1; x <= 10; x++ ) {
          printf( "%d ", square( x ) );
     }

    printf( "\n" );
    return 0;
}

int square( int y )
{
    return y * y;
}
```

break and continue

- **break** statement is used to immediately exit from the loop in which it is contained.
- **continue** statement is used to skip the remaining statements in the current **iteration** of the loop.
- Both the break and continue statements are useful in exiting either the current iteration or the entire loop when error conditions are encountered.

Example

```
sum=0;
for (k=1; k<=20; k++){
    scanf("%lf",&x);
    if (x > 10.0)
        break;
    sum += x;
}
```

Example

```
sum=0;
for (k=1; k<=20; k++){
    scanf("%lf",&x);
    if (x > 10.0)
        continue;
    sum += x;
}
```

Review Problem 1

- Write a program that takes user entered letter grades and counts how many A, B, C, D, and F are.

- The end of data entry will be indicated by an End Of File (EOF) character (done by either ctrl+Z or ctrl+D)

```c
/*Counting letter grades */
#include <stdio.h>

int main()
{
      int grade;
      int aCount = 0, bCount = 0, cCount = 0, dCount = 0, fCount = 0;

      printf ( "Enter the letter grades.\n" );
      printf ( "Enter the EOF character to end input.\n" );

      while ( ( grade = getchar() ) != EOF ) {

        switch ( grade ) {                  /* switch nested in while */

          case 'A': case 'a':               /* grade was uppercase A */
                ++aCount;                   /* or lowercase a */
                break;
          case 'B': case 'b':               /* grade was uppercase B */
                ++bCount;                   /* or lowercase b */
                break;
          case 'C': case 'c':               /* grade was uppercase C */
                ++cCount;                   /* or lowercase c */
                break;
          case 'D': case 'd':               /* grade was uppercase D */
                ++dCount;                   /* or lowercase d */
                break;
```

```
            case 'F': case 'f':                 /* grade was uppercase F */
                        ++fCount;               /* or lowercase f */
                      break;
                      case '\n': case ' ':      /* ignore these in input */
                      break;
                      default:                  /* catch all other characters */
                        printf( "Incorrect letter grade entered." );
                        printf( " Enter a new grade.\n" );
                        break;
                      }
                }
printf( "\nTotals for each letter grade are:\n" );
printf( "A: %d\n", aCount );
printf( "B: %d\n", bCount );
printf( "C: %d\n", cCount );
printf( "D: %d\n", dCount );
printf( "F: %d\n", fCount );
return 0;
}
```

Review Problem 2

- Generate a conversion table for converting degrees to radians. The degree values start at 0°, increment by 10°, and go through 360°.
  a) Use while structure
  b) Use do/while structure
  c) Use for structure

- **Pseudocode**

```
main:       set degrees to zero
            while degrees ≤ 360
                        convert degrees to radians
                        print degrees, radians
                        add 10 to degrees
```

```c
/* This program prints a degree-to-radian table */
/* using a while loop structure */

#include <stdio.h>
#include <stdlib.h>
#define PI 3.141593

main()
{
        int degrees=0;
        double radians;

        printf("Degrees to Radians \n");

        while (degrees <= 360)
        {
                radians = degrees*PI/180;
                printf("%6i %9.6f \n", degrees, radians);
                degrees += 10;
        }

        return EXIT_SUCCESS;
}
```

```c
/* This program prints a degree-to-radian table                    */
/* using a do/while loop structure                                 */

#include <stdio.h>
#include <stdlib.h>
#define PI 3.141593

main()
{
        int degrees=0;
        double radians;

        printf("Degrees to Radians \n");

        do
        {
                radians = degrees*PI/180;
                printf("%6i %9.6f \n", degrees, radians);
                degrees += 10;
        } while (degrees <= 360);

        return EXIT_SUCCESS;
}
```

```c
/* This program prints a degree-to-radian table                    */
/* using a for loop structure                                      */

#include <stdio.h>
#include <stdlib.h>
#define PI 3.141593

main()
{
        int degrees;
        double radians;

        printf("Degrees to Radians \n");

        for (degrees=0; degrees<=360; degrees+=10)
        {
                radians = degrees*PI/180;
                printf("%6i %9.6f \n", degrees, radians);
        }

        return EXIT_SUCCESS;
}
```

Review Problem 3

- Write a C program that reads a positive integer number from standard input, and finds the integer numbers between 1 and 1000, whose digits' sum are equal to the entered number, and prints the numbers to the standard output.

- Sample output:

```
Given inter number is 25

Numbers between 1 and 1000, whose digits' sum are equal to 25:
799
889
898
979
988
997
```

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

int main()
{
    int i, num_in, num_out, first_digit, second_digit,
     third_digit, sum;
    printf("enter a positive integer number and I will
     find the\n");
    printf("integer numbers between 1 and 1000, sum
     whose digits are \n");
    printf("equal to the entered number\n\n");

    while(1)
    {
      printf("your number:");
      scanf("%d",&num_in);
      if (num_in<=27) break;
      else
      {
        printf("your number is beyond the limit\n");
        printf("please enter a number <= 27\n\n");
      }
    }
    for (i=1;i<=1000;i++)
    {
        if ((1<=i)&&(i<=9)&&(i==num_in))
            printf ("%d\n",i);

        else if((10<=i)&&(i<=99))
        {
            first_digit=i%10;
            second_digit=i/10;
            sum=first_digit+second_digit;
            if (sum==num_in) printf ("%d\n",i);
        }

        else if((100<=i)&&(i<=999))
        {
            first_digit=i%10;
            second_digit=(i/10)%10;
            third_digit=(i/100);
            sum=first_digit+second_digit+third_digit;
            if (sum==num_in) printf ("%d\n",i);
        }
        else if((i==1000)&&(num_in==1))
            printf ("%d\n",i);
    }
    return 0;
}
```