Structured Programming
CSE 103

Professor Dr. Mohammad Abu Yousuf

Control Statement

Branching

- THE ***if*** STATEMENT

Syntax:

if (*expression)*

statement

The ***expression*** must be placed in parentheses, as shown. In this form, the ***statement*** will be executed only if the expression has a nonzero value (i.e., if expression is true). If the expression has a value of zero (i.e., if expression is false), then the statement will be ignored.

The *statement* can be either simple or compound.

Example : 1) **if(x < *0)***

printf ("%f" , x) ;

2**) if (x <= 3.0) {**

y = 3 * pow(x, 2) ;

printf ("The output is: %f\n", y) ;

}

Branching

- If statement example:

```
1.#include<stdio.h>
2.int main(){
3.    int number=0;
4.    printf("Enter a number:");
5.    scanf("%d",&number);
6.    if(number%2==0){
7.        printf("%d is even number",number);
8.    }
9.    return 0;
10.}
```

Output:
Enter a number:4
4 is even number

```c
#include <stdio.h>
int main()
{
    int a, b, c;
     printf("Enter three numbers?");
    scanf("%d %d %d",&a,&b,&c);
    if(a>b && a>c)
    {
        printf("%d is largest",a);
    }
    if(b>a  && b > c)
    {
        printf("%d is largest",b);
    }
    if(c>a && c>b)
    {
        printf("%d is largest",c);
    }
    if(a == b && a == c)
    {
        printf("All are equal");
    }
}
```

Program to find the largest number of the three.

Enter three numbers?
12 23 34
34 is largest

Branching

- THE ***if - else*** STATEMENT

  Syntax:

  if *(expression)*

  statement 1

  else

  statement 2

  If the expression has a nonzero value (i.e., if expression is true), then statement1 will be executed. Otherwise (i.e., if expression is false), statement2 will be executed.

- Example: **if (x <= 3)**

  y = 3 * pow(x, 2) ;

  else

  y = 2 * pow((x - 3) , 2) ;

Branching

If-else statement example

```
1.#include<stdio.h>
2.int main(){
3.    int number=0;
4.    printf("enter a number:");
5.    scanf("%d",&number);
6.    if(number%2==0){
7.        printf("%d is even number",number);
8.    }
9.    else{
0.      printf("%d is odd number",number);
1.   }
2.   return 0;
3.}
```

Output:
enter a number: 4
4 is even number

enter a number: 5
5 is odd number

Branching

- THE ***if - else if – else*** STATEMENT
  Syntax:

```
if (expression1)
        statement 1
else if  (expression2)
        statement 2
else if  (expression3)
        statement 3
else
        statement 4
```

Branching

- ***if - else if – else*** Example:

```
#include <stdio.h>
int main ()
{
        int x,y;
        printf ("\nInput an integer value for x: ");
        scanf ("%d", &x);
        printf ("\nInput an integer value for y: ");
        scanf ("%d",&y);
        if (x==y)
                printf ("x is equal to y\n");
        else if (x > y)
                printf ("x is greater than y\n");
        else
                printf ("x is smaller than y\n");
        return 0;
}
```

```c
#include<stdio.h>
int main(){
    int number=0;
    printf("enter a number:");
    scanf("%d",&number);
    if(number==10){
        printf("number is equals to 10");
    }
    else if(number==50){
        printf("number is equal to 50");
    }
    else if(number==100){
        printf("number is equal to 100");
    }
    else{
        printf("number is not equal to 10, 50 or 100");
    }
    return 0;
}
```

Output:

enter a number: 4
number is not equal to 10, 50 or 100

enter a number: 50
number is equal to 50

```c
#include <stdio.h>
int main()
{
    int marks;
    printf("Enter your marks?");
    scanf("%d",&marks);
    if(marks > 85 && marks <= 100)
    {
        printf("Congrats ! you scored grade A ...");

    }
    else if (marks > 60 && marks <= 85)
    {
        printf("You scored grade B + ...");
    }
    else if (marks > 40 && marks <= 60)
    {
        printf("You scored grade B ...");
    }
    else if (marks > 30 && marks <= 40)
    {
        printf("You scored grade C ...");
    }
    else
    {
        printf("Sorry you are fail ...");
    }
```

Program to calculate the grade of the student according to the specified marks

Output:

Enter your marks? 10
Sorry you are fail ...

Enter your marks? 40
You scored grade C ...

Enter your marks? 90
Congrats ! you scored grade A ...

Branching

- Nested if statement-

  Syntax:

  if e1 {

  if *e2 s1*

  }

  else *s2*

Nested If statement example

```
#include <stdio.h>
int main() {
    int number1, number2;
    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);
    if (number1 >= number2) {
      if (number1 == number2) {
        printf("Result: %d = %d",number1,number2);
      }
      else {
        printf("Result: %d > %d", number1, number2);
      }
    }
    else {
        printf("Result: %d < %d",number1, number2);
    }
```

LOOPING : THE ***while*** STATEMENT

- THE ***while*** STATEMENT

  The while statement is used to carry out looping operations, in which a group of statements is executed repeatedly, until some condition has been satisfied.

  Syntax:

  while (expression)

  statement

  The ***statement will be executed repeatedly, as long as the expression is true (i.e., as long expression*** has a nonzero value). This statement can be simple or compound, though it is usually a compound statement.

LOOPING : THE ***while*** STATEMENT

While statement example1:

- Suppose we want to display the consecutive digits 0, 1, 2, . . . ,9, with one digit on each line. This can be accomplished with the following program.

```
/* display the integers 0 through 9 */
#include  <stdio.h>
main ( )
 {
       int digit = 0;
       while (digit <= 9) {
             printf ( "%d\n", digit) ;
             ++digit;
        }
}
```

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

LOOPING : THE ***while*** STATEMENT

- While statement example 2 : Averaging list of numbers

```
#include <stdio.h>

main()

{
    int n, count = 1;
    float x, average, sum = 0;

    /* initialize and read in a value for n */
    printf("How many numbers? ");
    scanf("%d", &n);

    /* read in the numbers */
    while (count <= n)  {
          printf("x = ");
          scanf("%f", &x);
          sum += x;
          ++count;
    }

    /* calculate the average and display the answer */
    average = sum/n;
    printf("\nThe average is %f\n", average);
}
```

LOOPING : THE ***while*** STATEMENT

- **While statement example2 :**

  Output of previous slide program

```
How many numbers? 6
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6

The average is 3.500000
```

LOOPING : THE ***do-while*** STATEMENT

- Syntax:

  do

  statement

  while (expression);

  The statement will be executed repeatedly, as long as the value of expression is true (i.e., is nonzero).

  Notice that statement will always be executed at least once, since the test for repetition does not occur until the end of the first pass through the loop.

What is the difference between while and do-while ?

LOOPING : THE ***do-while*** STATEMENT

- **Consecutive Integer Quantities**

```
/* display the integers 0 through 9 */
#include <stdio.h>
main ( )
 {
       int digit = 0;
       do {
             printf ( "%d\n", digit++) ;
        }
        while (digit <= 9);
}
```

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

LOOPING: THE *for* STATEMENT

- **Syntax:**

for (***expression 1; expression 2; expression 3)***

{

statement

}

where expression 1 is used to initialize some parameter (called an index) that controls the looping action.

Expression 2 represents a condition that must be true for the loop to continue execution.

Expression3 is used to alter the value of the parameter initially assigned by expression 1. Typically, expression 1 is an assignment expression, expression 2 is a logical expression and expression 3 is a unary expression or an assignment expression.

LOOPING: THE *for* STATEMENT

- **Consecutive Integer Quantities**

```
/* display the numbers 0 through 9 */
#include <stdio.h>
main()
{
 int digit;
 for (digit = 0; digit <= 9; ++digit)
        printf('%d\n", digit);
}
```

LOOPING: THE *for* STATEMENT

- Averaging a list of numbers

```
/* calculate the average of n numbers */

#include <stdio.h>

main()

{
    int n, count;
    float x, average, sum = 0;

    /* initialize and read in a value for n */
    printf("How many numbers? ");
    scanf("%d", &n);
    /* read in the numbers */
    for (count = 1; count <= n; ++count)  {
        printf("x = ");
        scanf("%f", &x);
        sum += x;
    }

    /* calculate the average and display the answer */
    average = sum/n;
    printf("\nThe average is %f\n", average);
}
```

Nested Loops in C

- Any number of loops can be defined inside another loop, i.e., there is no restriction for defining any number of loops. The nesting level can be defined at n times.

```
1.Outer_loop
2.{
3.   Inner_loop
4.   {
5.        // inner loop statements.
6.   }
7.      // outer loop statements.
8.}
```

```
1.for (initialization; condition; update)
2.{
3.    for(initialization; condition; update)
4.    {
5.          // inner loop statements.
6.    }
7.    // outer loop statements.
8.}
```

Nested Loops in C

```
1.#include <stdio.h>
2.int main()
3.{
4.   int n;// variable declaration
5.   printf("Enter the value of n :");
6.   scanf(“%d”, &n);
7.   for(int i=1;i<=n;i++)  // outer loop
8.   {
9.      for(int j=1;j<=10;j++)  // inner loop
0.      {
1.         printf("%d\t",(i*j)); // printing the value.
2.      }
3.      printf("\n");
4.   }
```

```
Enter the value of n : 3
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
```

Infinite Loop in C

- An infinite loop is a looping construct that does not terminate the loop and executes the loop forever. It is also called an **indefinite** loop.

```
1.for(; ;)
2.{
3.    // body of the for loop.
4.}
```

```
1.#include <stdio.h>
2.int main()
3.{
4.   for(;;)
5.   {
6.     printf("Hello CSE");
7.   }
8.return 0;
9.}
```

Infinite Loop in C

```
1.while(1)
2.{
3.   // body of the loop..
4.}
```

```
1.do
2.{
3.    // body of the loop..
4.}while(1);
```

```
1.#include <stdio.h>
2.int main()
3.{
4.  int i=0;
5.  while(1)
6.  {
7.      i++;
8.      printf("i is :%d",i);
9.  }
10.return 0;
11.}
```

Thank you