Structured Programming
CSE 103

Professor Dr. Mohammad Abu Yousuf

THE switch STATEMENT

- The switch statement causes a particular group of statements to be chosen from several available groups. The selection is based upon the current value of an expression which is included within the switch statement.
- The general form of the switch statement is

  switch *(expression)*

  statement

  where *expression results in an integer value. Note that expression may also be of type char, since* individual characters have equivalent integer values.

  The embedded statement is generally a compound statement that specifies alternate courses of action.

THE switch STATEMENT (Cont…)

- For each alternative, the first statement within the group must be preceded by one or more case labels (also called *case prefixes). The case labels identify the different groups of statements (i.e., the different* alternatives) and distinguish then from one another. The case labels must therefore be unique within a given switch statement.
- In general terms, each group of statements is written as

  case *expression :*

  statement 1

  statement 2

  . . . . .

  statement n

THE switch STATEMENT (Cont…)

- or, when multiple case labels are required,

  case ***expression 1 :***

  case ***expression 2 :***

  . . . . .

  case ***expression m :***

  statement 1

  statement 2

  . . . …

  statement n

  where *expression 1, expression 2, . . . , expression m represent constant, integer-valued* expressions. Usually, each of these expressions will be written as either an integer constant or a character constant. Each individual *statement following the case labels may be either simple or complex.*

```
1.#include<stdio.h>
2.int main(){
3.    int number=0;
4.    printf("enter a number:");
5.    scanf("%d",&number);
6.    switch(number){
7.       case 10:
8.          printf("number is equals to 10");
9.          break;
0.     case 50:
1.          printf("number is equal to 50");
2.          break;
3.     case 100:
4.          printf("number is equal to 100");
5.          break;
6.     default:
7.          printf("number is not equal to 10, 50 or 100");
8.    }
9.  return 0;
0. }
```

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

THE switch STATEMENT (Cont…)

- Example: In this example, **choice is assumed to be a char-type** variable.

```
switch (choice = getchar())  {

case 'r':
case 'R':
      printf("RED");
      break;
case 'w':
case 'W':
     printf("WHITE");
     break;

case 'b':
case 'B':
     printf("BLUE");
}
```

THE switch STATEMENT (Cont…)

- One of the labeled groups of statements within the switch statement may be labeled default. This group will be selected if none of the case labels matches the value of the *expression.*

```
switch (choice = toupper(getchar()))  {

case 'R':
     printf("RED");
     break;

case 'W':
     printf("WHITE");
     break;

case 'B':
     printf("BLUE");
     break;

default:
     printf("ERROR");
}
```

THE break STATEMENT

- The break statement is used to terminate loops or to exit from a switch. It can be used within a for, while, do -while, or switch statement.
- The break statement is written simply **as**

  break;

  without any embedded expressions or statements.
- If a break statement is included in a while, do - while or for loop, then control will immediately be transferred out of the loop when the break statement is encountered.

THE break STATEMENT (Cont..)

```
1.#include<stdio.h>
2.#include<stdlib.h>
3.void main ()
4.{
5.    int i;
6.    for(i = 0; i<10; i++)
7.    {
8.        printf("%d ",i);
9.        if(i == 5)
10.       break;
11.   }
12.   printf("came outside of loop i = %d",i);
13.
14.}
```

Output:
0 1 2 3 4 5 came outside of loop i = 5

THE break STATEMENT (Cont..)

- In such case, it breaks only the inner loop, but not outer loop.

```
1.#include<stdio.h>
2.int main(){
3.   int i=1,j=1;//initializing a local variable
4.   for(i=1;i<=3;i++){
5.      for(j=1;j<=3;j++){
6.         printf("%d  %d\n",i,j);
7.         if(i==2 && j==2){
8.            break;//will break loop of j only
9.         }
10.    }//end of inner for loop
11.  } //end of outer for loop
12.  return 0;
13.}
```

Output:

```
1  1
1  2
1  3
2  1
2  2
3  1
3  2
3  3
```

As you can see the output on the console, 2 3 is not printed because there is a break statement after printing i==2 and j==2. But 3 1, 3 2 and 3 3 are printed because the break statement is used to break the inner loop only.

THE break STATEMENT (Cont..)

- Find the output:

```
1.#include<stdio.h>
2.void main ()
3.{
4.    int i = 0;
5.    while(1)
6.    {
7.        printf("%d  ",i);
8.        i++;
9.        if(i == 10)
10.       break;
11.   }
12.   printf("came out of while loop");
13.}
```

THE break STATEMENT (Cont..)

Output:
0 1 2 3 4 5 6 7 8 9 came out of while loop

THE continue STATEMENT

- The ***continue statement*** is used to *bypass the remainder of the current pass through a loop. The loop does not terminate when a continue statement is encountered. Rather, the remaining loop statements are skipped* and the computation proceeds directly to the next pass through the loop. (Note the distinction between **continue and break.)**
- The continue statement can be included within **a while, a do - while** or **a for statement.**
- It is written simply as

  continue;

  without any embedded statements or expressions.

THE continue STATEMENT

```
1.#include<stdio.h>
2.int main(){
3.   int i=1;//initializing a local variable
4.   //starting a loop from 1 to 10
5.   for(i=1;i<=10;i++){
6.     if(i==5){//if value of i is equal to 5, it will continue the loop
7.           continue;
8.       }
9.    printf("%d \n",i);
10.  }//end of for loop
11.  return 0;
12.}
```

Output:
1
2
3
4
6
7
8
9
10

THE continue STATEMENT

- Find the output:

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

THE continue STATEMENT

- Output:

  Infinte loop

- Example *continue*:

```
#include <stdio.h>

main()

{
    int i = 0, x = 0;

    for (i = 1; i < 10; ++i)   {
        if (i % 2 == 1)
            x += i;
        else
            x--;
        printf("%d ", x);
        continue;
    }
    printf("\nx = %d", x);
}
```

THE continue STATEMENT

Answer: **1 0 3 2 7 6 13 12 21**

X=21

- Example *break:*

```
#include <stdio.h>

main()

{
    int i = 0, x = 0;

    for (i = 1; i < 10; ++i)   {
        if (i % 2 == 1)
            x += i;
        else
            x--;
        printf("%d ", x);
        break;
    }
    printf("\nx = %d", x);
}
```

- Answer: **1**

  x = 1

C goto statement

- The "**goto**" statement is known as jump statement in C. As the name suggests, "**goto**" is used to transfer the program control to a predefined label.

Syntax:

```
label:
//some part of the code;
goto label;
```

```
1.#include <stdio.h>
2.int main()
3.{
4.  int num,i=1;
5.  printf("Enter the number :");
6.  scanf("%d",&num);
7.  table:
8.  printf("%d x %d = %d\n",num,i,num*i);
9.  i++;
10. if(i<=10)
11.     goto table;
12.}
```

Output:

```
Enter the number: 10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

Thank you