Alternative Control Structures

Outline

- switch statement
- do-while statement
- for Statement

switch Statement

```
switch (IntegralExpression condition)
{
   case Constant1:
        Statement1;
        break;
   case Constant2:
        Statement2;
        break;
   case Constant3:
        Statement3;
        break;
   default:
        Default statement;
 }
```


Switch Statement

- the value of *IntegralExpression* (of char, short, int, long or enum type) determines which branch is executed
- case labels are constant (possibly named) integral expressions. Several case labels can precede a statement

Control in Switch Statement

- control branches to the statement following the case label that matches the value of *IntegralExpression*. Control proceeds through all remaining statements, including the default, unless redirected with break
- if no case label matches the value of *IntegralExpression*, control branches to the default label, if present--otherwise control passes to the statement following the entire switch statement
- forgetting to use break can cause logical errors because after a branch is taken, control proceeds sequentially until either break or the end of the switch statement occurs

Example 1: Binary Operations

```
char Operator;
int Number1, Number2;
cout<<"Enter two integers -->";
cin>>Number1>>Number2;
cout<<"Enter an operator -->";
cin>>Operator;
switch (Operator)
{
   case '+':
        cout<<"The sum is "<<Number1+Number2;
        break;
   case '-':
        cout<<"The difference is "<<Number1-Number2;
        break;
   case '/':
        cout<<"The quotient is "<<Number1/Number2;
        break;
   case '%':
        cout<<"The remainder is "<<Number1%Number2;
        break;
   default:
        cout<<"Invalid operator!!!!";
 }
```


Example 2: Assigning Grades

```
if (Average >= 90)
   cout<<"Your grade is an A";
else if (Average >= 80)
        cout<<"Your grade is a B";
     else if (Average >= 70)
             cout<<"Your grade is a C";
          else if (Average >= 60)
                  cout<<"Your grade is a D";
               else
                  cout<<"Your grade is an F";
```


Example 3: Compute Weight

```
float   weightInPounds = 165.8;
char    weightUnit ;
     .  .  .
//  user enters letter for desired weightUnit
switch  ( weightUnit )
{
      case 'P':
      case 'p':
            cout << weightInPounds << " pounds " << endl;
            break;
      case 'O':
      case 'o':
            cout << 16.0 * weightInPounds << " ounces " << endl;
              break;
      case 'K':
      case 'k':
            cout << weightInPounds / 2.2 << " kilos " << endl;
              break;
      case 'G':
      case 'g':
            cout << 454.0 * weightInPounds << " grams " << endl;
              break;
      default:
            cout << "That unit is not handled! " << endl;
}
```

do-while Loop

```
do
{
    Statement1;
    Statement2;
} while (loop condition);
```

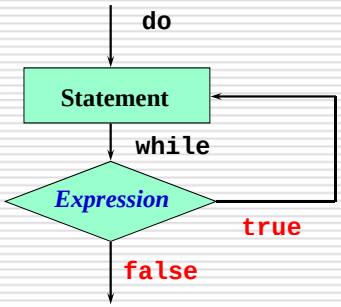
do-while is a looping control structure in which the loop condition is tested after each iteration of the loop.

do-while Loop vs. while Loop

- POST-TEST loop (exit-condition)
- The looping condition is tested after executing the loop body.
- Loop body is always executed at least once.

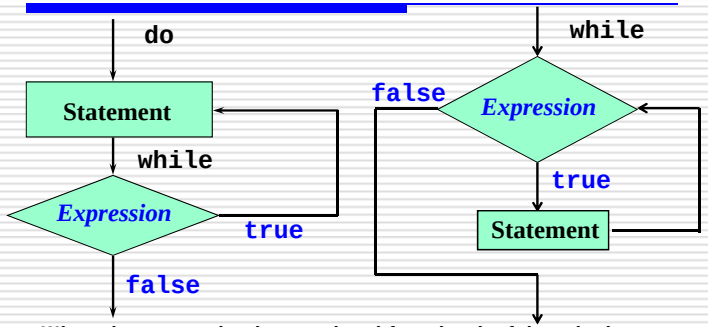
- PRE-TEST loop (entry-condition)
- The looping condition is tested before executing the loop body.
- Loop body may not be executed at all.

Do-While Loop



When the expression is tested and found to be false, the loop is exited and control passes to the statement that follows the do-while statement.

Do-While Loop



When the expression is tested and found to be false, the loop is exited and control passes to the statement that follows the do-while statement.

Example 1

```
//Response = 'y';
do
{
   cout<<"Enter your weight";
   cin>>Weight;
   if  (Weight < 300)
      cout<<"You are okay."
   else
      cout<<"You are over weight.";
   cout<<"Do you want to try again?";
   cin>>Response;
}   while (Response == 'y');
```


Example 2: Find the Sum of 10 integers

```
Size = 10;
Count = 1;
Sum = 0;
do
{
   cout<<"Enter an integer";
   cin>>INumber;
   Sum = Sum + INumber;
   Count = Count + 1;
}  while (Count <= Size);
cout<<"The sum is "<<Sum;
```


Example 3: Find Maximum number of 10 integers

```
Count = 1;
cout<<"Enter an integer";
cin>>INumber;
MaxNumber = INumber;
do
{
   cout<<"Enter an integer";
   cin>>INumber;
   if (MaxNumber < INumber)
      MaxNumber = INumber;
   Count = Count + 1;
} while (Count < Size);
cout<<"The maximum number is "<<MaxNumber;
```


for Statement

```
for (Initialization Statement;
      Loop condition; Update statement)
   Statement1;
```

for-loop is a count controlled loop and a compact version of **while-loop**.

- **an initialization statement**
- **A loop condition to test for continuing**
- **an update to execute after each iteration of the body**

Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num ?

Output

Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 1

Output

Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 1

Output

Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 1

Output

```
1 Indian
```


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 2

Output

```
1 Indian
```


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 2

Output

```
1 Indian
```


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 2

Output

```
1 Indian
2 Indian
```

Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 3

Output

```
1 Indian
2 Indian
```


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 3

Output

```
1 Indian
2 Indian
```


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 3

Output

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


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 4

Output

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

Example

```
int num;
for(  num = 1; num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 4

Output

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


Example

```
int num;
for(  num = 1;  num <= 3;  num++   )
{
    cout << num << " Indian" << endl;
}
cout <<"Done" << endl;
```

num 4

Output

```
1 Indian
2 Indian
3 Indian
Done
```


Example 1

```
int Count, Sum;
Sum = 0;
for (Count = 1; Count <= 10; Count++)
     Sum = Sum + Count;
```

```
int Sum;
Sum = 0;
for (int Count = 1; Count <= 10; Count++)
     Sum = Sum + Count;
```


Example 2:Find Sum of 10 Integers

```
Size = 10;
Sum = 0;
for (int Count = 1; Count <= Size; Count++)
{
   cout<<"Enter an integer";
   cin>>INumber;
   Sum = Sum + INumber;
}
cout<<"The sum is "<<Sum;
```

Example 3: Find Maximum number of 10 integers

```
Size = 10;
cout<<"Enter an integer";
cin>>INumber;
MaxNumber = INumber;
for (int Count = 1; Count <= Size; Count++)
{
   cout<<"Enter an integer";
   cin>>INumber;
   if (MaxNumber < INumber)
      MaxNumber = INumber;
}
cout<<"The maximum number is "<<MaxNumber;
```


Example 4

```
int count;

for(count = 4; count > 0; count--)
{
    cout << count << endl;

}
cout << "Done" << endl;
```

Output:

4
3
2
1
Done

What is output?

```
int count;

for(count = 0; count < 10; count++)
{
  cout << "*" << endl;
}
```

Output

****

What is output?

```
int count;

for(count = 0; count < 10; count++);
{
  cout << "*" << endl;
}
```

Output

*

OUTPUT

- no output from the **for** loop in the previous example. Why?
- the **;** right after the ( ) means that the body statement is a null statement
- in general, the Body of the for loop is whatever statement immediately follows the ( )
- that statement can be a single statement, a block, or a null statement
- actually, the code outputs one * after the loop completes its counting to 10

Break Statement

- **break** statement can be used with Switch or any of the 3 looping structures
- it causes an immediate exit from the **switch**, **while**, **do-While**, or **for** statement in which it appears
- if the **break** is inside nested structures, control exits only the innermost structure containing it

Example of Nested for-loop

Write a code segment to output the following:

```
**********
**********
**********
**********
```

```
for (int outer = 1; outer <= 4; outer++)
{
   for (int inner = 1; inner <= 4; inner++)
      cout<<'*';
   cout<<endl;
}
```


Example of Nested for-loop

Write a code segment to output the following:

```
0123456789
*********
 ********
  *******
   ******
    *****
     ****
      ***
       **
        *
```

```
numStar = 10;
cout<<"0123456789"<<endl;

for (int outer = 0; outer < numStar; outer++)
{
   for (int blank = 0; blank < outer; blank++)
      cout<<' ';
   for (int star = numStar - outer; star > 0; star--)
      cout<<'*';
   cout<<endl;
}
```

Example

Write a C++ program that inputs an integer and a character. The output is a triangle shape form by the character starting with one character on the first line, 3 on the second line, the last line will have the number of characters as the integer input. For example, the input is **11 *** the output is

```
          *
         ***
        *****
       *******
      *********
     ***********
```


Example of Nested for-loop

Write a code segment to output the following:
If the input 11 →

```
     11
     ***********
      *********
       *******
        *****
         ***
          *
```

If the input 4 →

```
     4
     *****
      ***
       *
```

```
cout<<"Enter a non-negative integer -->";
cin>>inputNumber;

if (inputNumber % 2 == 0)
   star = inputNumber + 1;

cout<<inputNumber<<endl;

for (int outer =1; outer <= star/2 + 1; outer++)
{
   for (int inner = 1; inner <= star; inner = inner - 2)
      cout<<'*';
   cout<<endl;
}
```


Guidelines for Choosing Looping Statement

- For a simple count-controlled loop, use the For statement
- For an event-controlled loop whose body always executes once, use of Do-While statement
- For an event-controlled loop about which nothing is known, use a While statement
- When in doubt, use a While statement