Control Structures: Selection Statement

Chapter 3
B. Ramamurthy

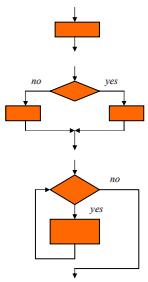
Introduction

"Two roads divulged in a yellow wood,
And sorry I could not travel both…."

—— Robert Frost
From "The Road Not Taken"

Structured Programming

- Sequence
- Selection
- Repetition



Decision Statements (selection)

- How to compare data values?
  Relational operators
- How to alter the sequence of program execution based on the result?
  if.. else statements
- How to deal with multiple choices?
  Switch statement

Example

- We will use "assigning letter grade based on percentage points" as an example to illustrate selection statement.

Percent Range	>=90	>=80 < 90	>=70 <80	>=60 <70	<50
Letter Grade	A	B	C	D	F

Relational Operators

Operator	Meaning
<	Less than ?
>	Greater than ?
>=	Greater than or equal to?
<=	Less than or equal to?
==	Equal to?
!=	Not Equal to ?

Logical Operators

!	not
&&	and
||	or

These operators are used to combine more than one condition forming a complex condition.

if Statement

- An if statement allows a program to choose whether or not to execute a following statement.
- Syntax:

if (**condition**)
  statement;

- Semantics:

condition is a Boolean expression: Something that evaluates to True or False.

If **condition** is true then execute the **statement** is executed.

The if statement : syntax

```
if(expression)
    statement;        //single statement executed
                      //if expression is true

if(expression)
{                     //statements inside {} are
                      //executed if expression is true
    statement1;
    statement2;
    ...
    statement n;
}
```


If -else Statement

- An if-else statement allows a program to do one thing if a condition is true and a different thing if the condition is false.
- Syntax:

if (condition)
  statement1
else
  statement2

- Statements to be executed for if and else can be a single statement or multiple statements enclosed in { }.

The if - else statement: syntax

```
if(expression)
   statement;
else
   statement;

if(expression)
   {
      statement block
   }
else
   {
      statement block
   }
```


The switch statement

```
switch(expression)
{
   case constant:
         statement(s);
         break;
   case constant:
         statement(s);
         break;
    /* default is optional*/
   default:
         statement(s);
}
```

The switch statement

- *Expression* must be of type integer or character
- The keyword **case** must be followed by a *constant*
- **break** statement is required unless you want all subsequent statements to be executed.

Practice!

Convert these nested **if/else** statements to a **switch** statement:

```
if (rank==1 || rank==2)
   cout << "Lower division \n";
else
{   if (rank==3 || rank==4)
        cout << "Upper division \n";
    else
    {   if (rank==5)
            cout << "Graduate student \n";
        else
            cout << "Invalid rank \n";
    }
}
```


Practice Solution!

```
switch(rank)

{
  case 1: case 2:
    cout << "Lower division \n";
    break;
  case 3: case 4:
    cout << "Upper division \n";
    break;
  case 5:
    cout << "Graduate student \n";
    break;
  default:
    cout << "Invalid rank \n";
}//end switch
```


Summary

- In many applications, choices are to be made depending on some conditions related to the problem. Selection or decision structures are used to model such situations.
- C++ supports the implementation of "selection" through the "if" and "switch" statements. In this discussion we looked at various forms of selection statements.