Department of Networks

First Year

Problem Solving and Programming 1

C++ Assignment Operators

- Assignment operators are used to assign values to variables.
- In the example below, we use the assignment operator (=) to assign the value 10 to a variable called x:

Example

```
int x = 10;
```

- The addition assignment operator (+=) adds a value to a variable:

Example

```
int x = 10;

x += 5;
```

A list of all assignment operators:

Operator	Example	Same As
=	X = 10	X = 10
+=	X += 3	X = X+3
-=	X -= 3	X = X-3
*=	X *= 3	X = X*3
/=	X /= 3	X = X/3
%=	X %= 3	X = X%3

C++ Comparison Operators

- Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	X > Y
<	Less than	X < Y
>=	Greater than or equal to	X>=Y
<=	Less than or equal to	X<=Y

C++ Logical Operators

- Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x > 1 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x > 10
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

C++ Math

- C++ has some default functions that allows you to perform mathematical tasks on numbers.

1. `max(x,y)`

- The **`max(x,y)`** function can be used to find the highest value of x and y:

Example

```
cout<<max(5, 10);
```

2. **`min(x,y)`**

- The **`min(x,y)`** function can be used to find the lowest value of x and y:

Example

```
R = min(5, 10);
```

- Other functions, such as **`sqrt`** (square root), **`round`** (rounds a number) and **`abs`** (absolute value), can be found in the <cmath>

3. **`sqrt(x)`**

- The sqrt(x) function returns the square root of x:

Example

```
M = sqrt(64);
```

4. **`abs(x)`**

- The **`abs(x)`** function returns the absolute (positive) value of x:

Example

```
cout<<abs(-4.7);
```

5. **`round(x)`**

- round() rounds a number to the nearest whole number:

Example

```
B = round(9.99);
```

Example: Implement the following equation in C++:

$$\boldsymbol{y = \sqrt{(x-4)}}$$

```
int x;
float y;
cin>>x ;
y = sqrt (x-4);
cout<<y;
```

C++ Conditional Statements

- C++ supports the usual logical conditions from mathematics:
  - Less than: a < b
  - Less than or equal to: a <= b
  - Greater than: a > b
  - Greater than or equal to: a >= b
  - Equal to a == b
  - Not Equal to: a != b
- You can use these conditions to perform different actions for different decisions.
- C++ has the following conditional statements:

– Use **`if`** to specify a block of code to be executed, if a specified condition is true

– Use **`else`** to specify a block of code to be executed, if the same condition is false

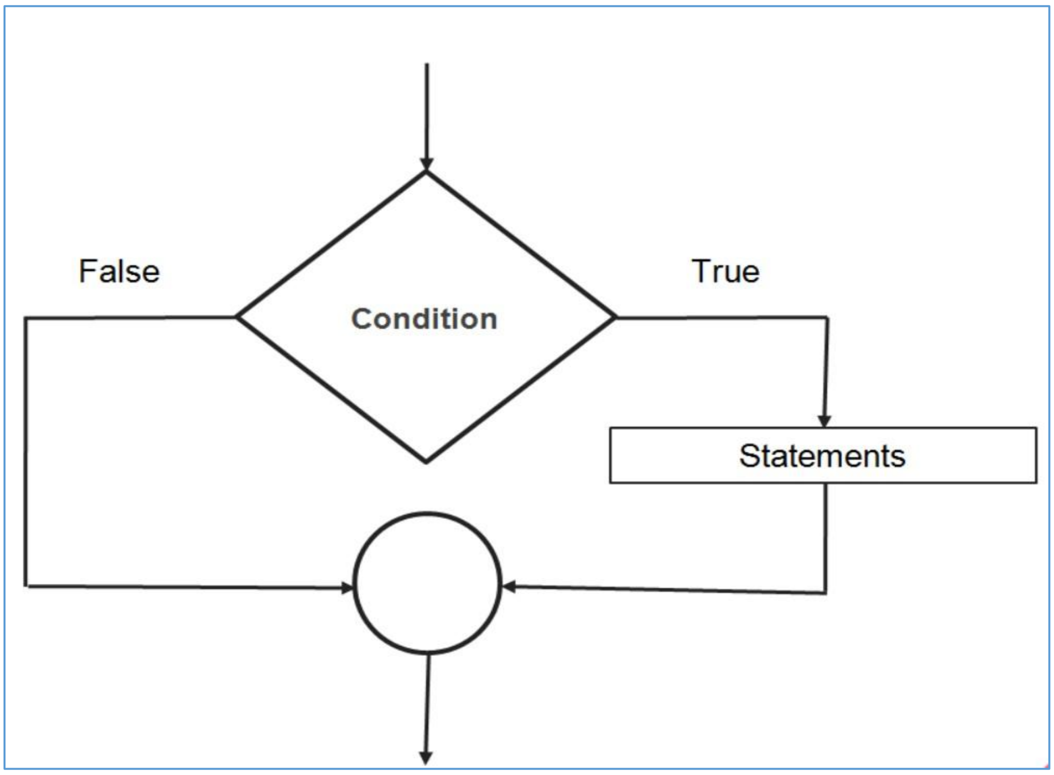
– Use **`else if`** to specify a new condition to test, if the first condition is false
– Use **`switch`** to specify many alternative blocks of code to be executed

The `if` Statement

- Use the **`if`** statement to specify a block of C++ code to be executed if a condition is True.

Syntax

```
  if (condition)

  {

    // block of code to be executed if the
condition is True

  }
```



- In the example below, we test two values to find out if 20 is greater than 18. If the condition is True, print some text:

Example

```
int x = 20;

int y = 18;

if (x > y)

{

  cout<<"x is greater than y";

}
```

Example explained

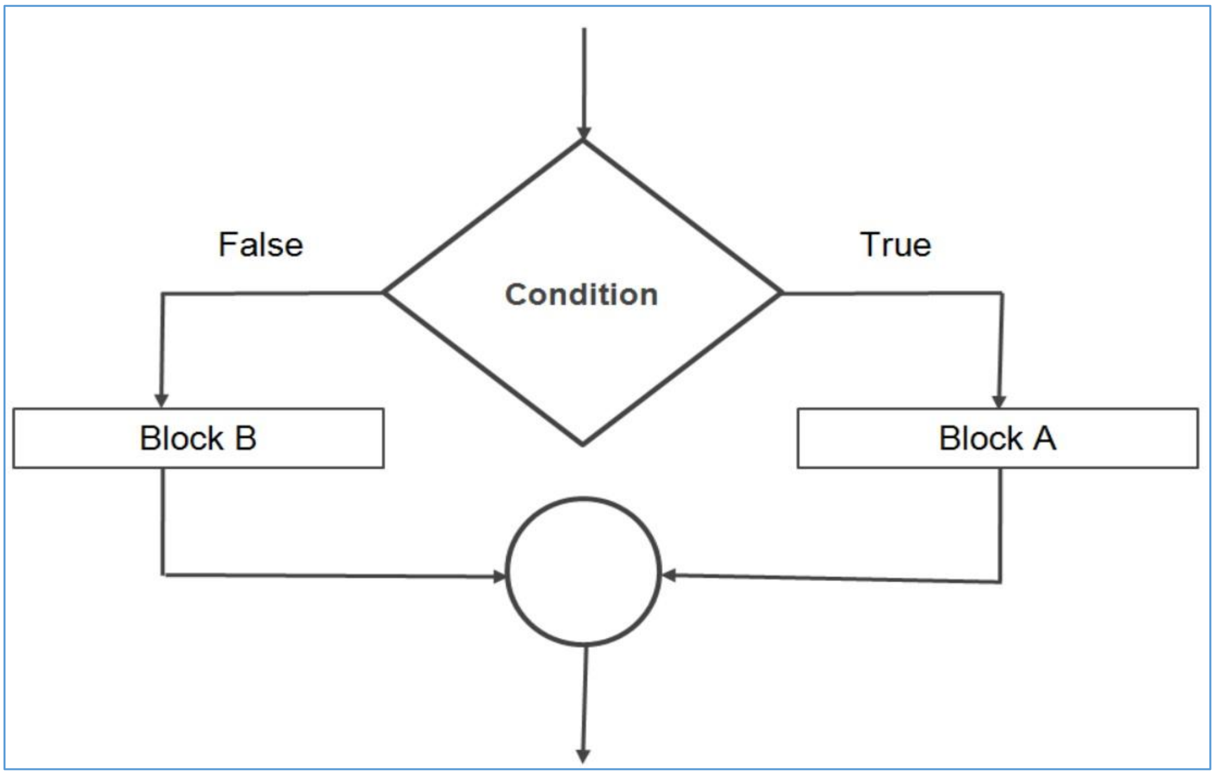
- In the example above we use two variables, x and y, to test whether x is greater than y (using the > operator). As x is 20, and y is 18, and we know that 20 is greater than 18, we print to the screen that "x is greater than y".

Example

```
int x;

int max = 0;

  cin>>x

if (x > max)

{

   max = x;

}
```

The else Statement

- Use the **`else`** statement to specify a block of code to be executed if the condition is False.



Syntax

```
  if (condition)

  {

    // block of code to be executed if the
condition is True

  }

  else

  {

    // block of code to be executed if the
condition is False

  }
```

Example 1

```
int time = 20;

if (time < 18)

{

  cout<<("Good day.");

}

else

{

  cout<<"Good evening.";

}// Outputs "Good evening."
```

Example explained

- In the example above, time (20) is greater than 18, so the condition is False. Because of this, we move on to the else condition and print to the screen "Good evening". If the time was less than 18, the program would print "Good day".

Example 2: How to check if a number is even or odd

```
int x;

cin>>x;

if(x%2==0)

{

  cout<<"Even";

}

else
```

```
{
  cout<<"Odd";
}
```

Example 3: How to check if a number is positive or negative

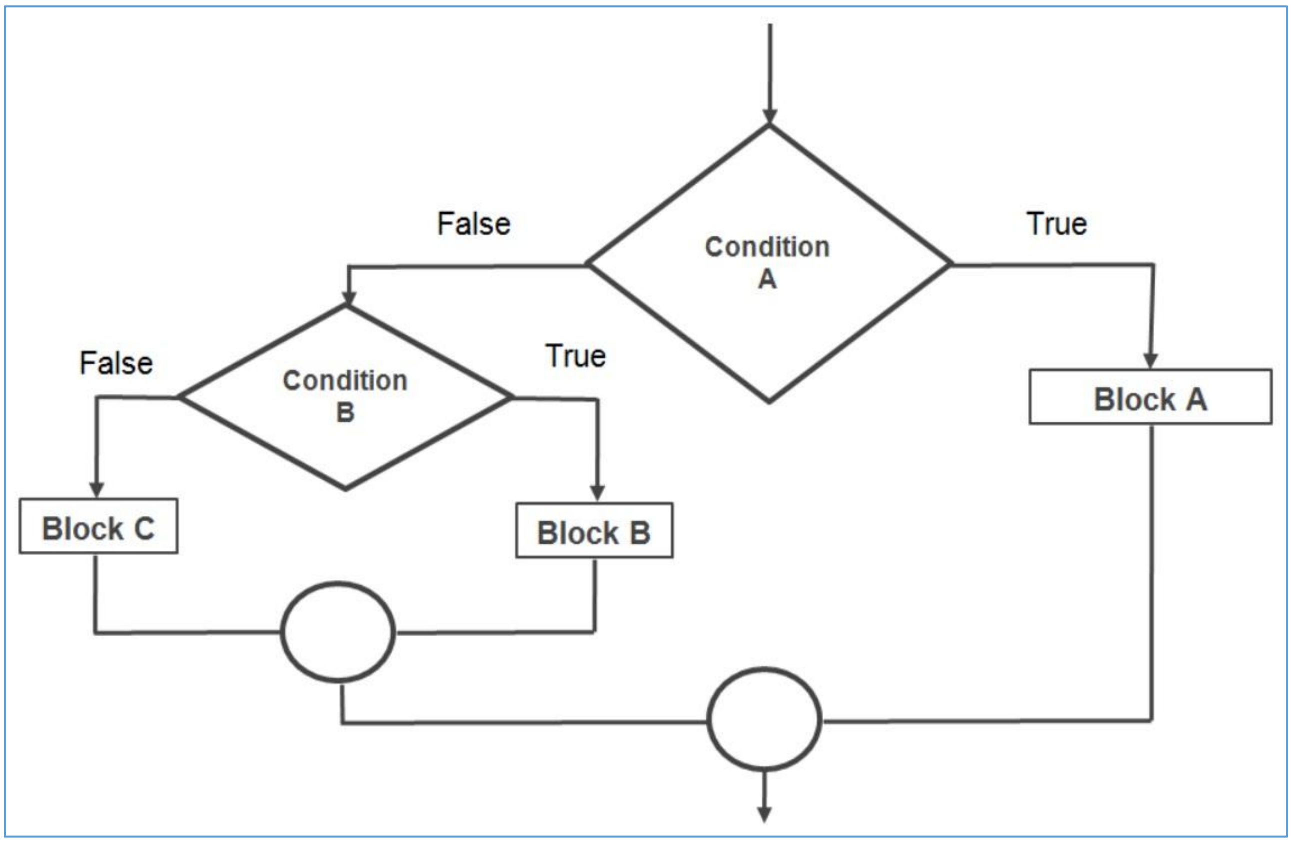
```
int x;
cin>>x;
if(x<0)
{
  cout<<"Negative";
}
else
{
  cout<<"Positive";
}
```

Example 4: How to calculate the profit and loss of a company

```
int income, cost;
int profit, loss;
cin>>income;
cin>>cost;
if(income>=cost)
{
    profit = income - cost;
    cout<<"Profit = "<< profit;
}
else
{
    loss = cost - income;
    cout<<"Loss = " << loss;
}
```

The else if Statement

- Use the **`else if`** statement to specify a new condition if the first condition is False.



Syntax

```
  if (condition1)
  {
    // block of code to be executed if condition1
is True
  }
  else if (condition2)

  {

    // block of code to be executed if the
condition1 is false and condition2 is True

  }

  else

  {

    // block of code to be executed if the
condition1 is false and condition2 is False

  }
```

Example

```
    int time = 22;
    if (time < 10)
    {
      cout<<"Good morning.";
    }
    else if (time < 20)
    {
      cout<<"Good day.";
    }
    else
    {
      cout<<"Good evening.";
    }// Outputs "Good evening."
```

Example explained

- In the example above, time (22) is greater than 10, so the first condition is False. The next condition, in the else if statement, is also False, so we move on to the else condition since condition1 and condition2 is both False - and print to the screen "Good evening". However, if the time was 14, our program would print "Good day."

C++ Switch Statements

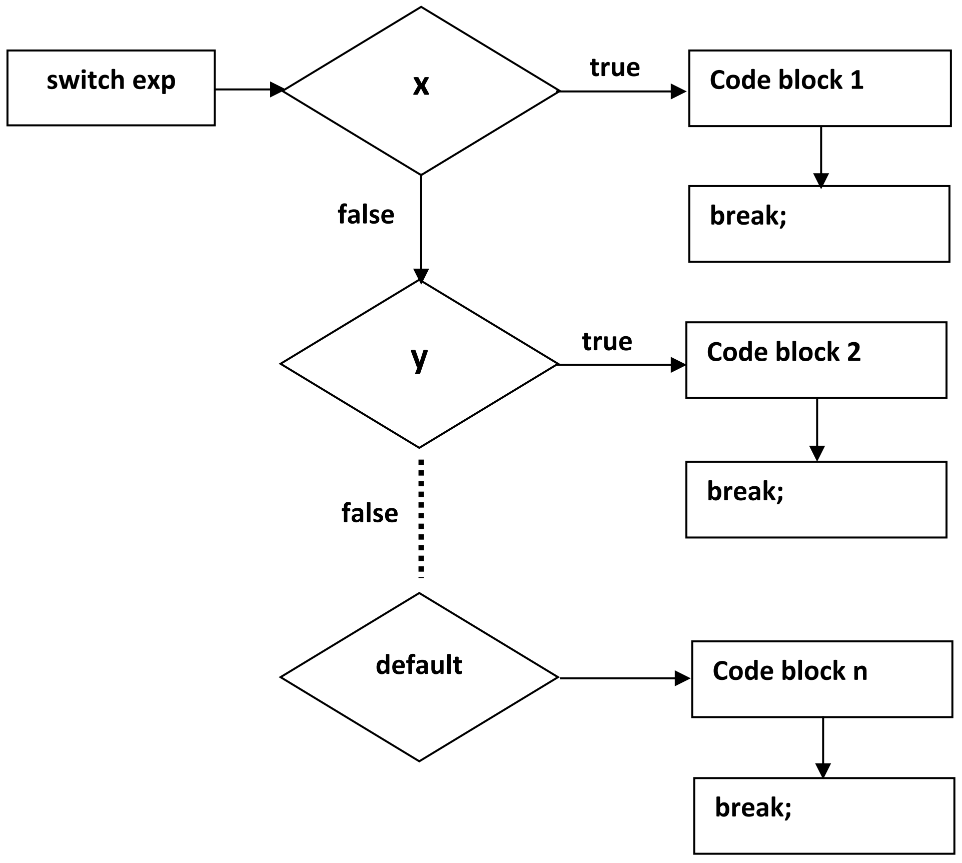
- Use the switch statement to select one of many code blocks to be executed.

Syntax

```
switch(expression)
{
  case x:
    // code block 1
    break;

  case y:
    // code block 2
    break;
.
.
.
  default:
    // code block 3
    break;
}
```

This is how it works:

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break and default keywords will be described later.



The example below uses the weekday number to calculate the weekday name:

Example

```
int day = 4;
switch (day)
```

```
{
  case 1:
    cout<<"Monday";
    break;
  case 2:
    cout<<"Tuesday";
    break;
  case 3:
    cout<<"Wednesday";
    break;
  case 4:
    cout<<"Thursday";
    break;
  case 5:
   cout<<"Friday";
    break;
  case 6:
    cout<<"Saturday";
    break;
  case 7:
    cout<<"Sunday";
    break;
}// Outputs "Thursday" (day 4)
```

The break Keyword

- When C++ reaches a `break` keyword, it breaks out of the switch block.

- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break. There is no need for more testing.
- A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.

The `default` Keyword

- The **`default`** keyword is optional and specifies some code to run if there is no case match:

Example

```
int day = 4;
switch (day)
{
  case 6:
    Cout<<("Today is Saturday.");
    break;
  case 7:
    Cout<<("Today is Sunday.");
    break;
  default:
    Cout<<("Looking forward to the Weekend.");
    break;
}// Outputs "Looking forward to the Weekend."
--------------------------------------
```

Homework:

1. Write a C++ program to enter an integer number from the keyboard and check if the first digit is divisible by 3.
2. Write a C++ program to enter a float number **`x`** and check whether **`x`** is odd or even **<u>without</u>** using the mod operator (%).
3. Write a C++ to check whether a character is Uppercase or Lowercase. (**Hint:** To read a character from the keyboard use:

   ```
   char ch;

   cin>>ch;
   ```

4. Write a C++ to enter a character from the keyboard and check whether the character is a digit or not.
5. Write a C++ program to enter a character from the keyboard and check whether the character is alphabet or not.
6. Write a C++ program to enter month number and print the month name.

   Example:

   input

   input: 7

   output

   July

   Example:

input

2

output

February