

* In this EET 340 course, we will be focused on assembly language. We will convert high level language (C++) into assembly language. The lab assignment will be conducted using a simulator.

* You just need to know the basics (simple programs) of C++ programming. If you do not have any prior knowledge on programming, that will be okay. I will teach and review C++ programming in a few lectures. You need to focus on those lectures. It will be fun to learn.

* Most of the high-level languages (C, C++, Java, Matlab) are quite similar. If you already know one language (C/ Java/ Matlab/others), you can learn/understand C++ language very easily.

Part 1: Running a simple program

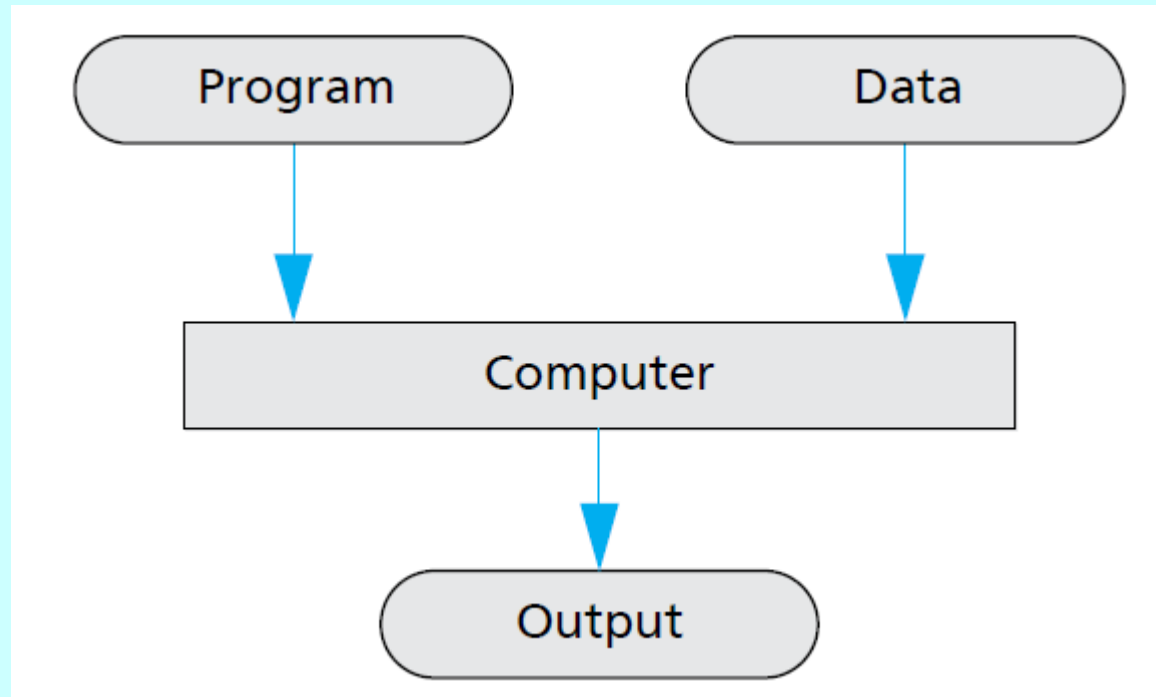
Computer Systems

- A Computer program is a process which is a sequence of instructions or tasks. A computer program tells a computer , in minute detail, the sequence of steps that are needed to fulfill a task.

Running/Executing a Program

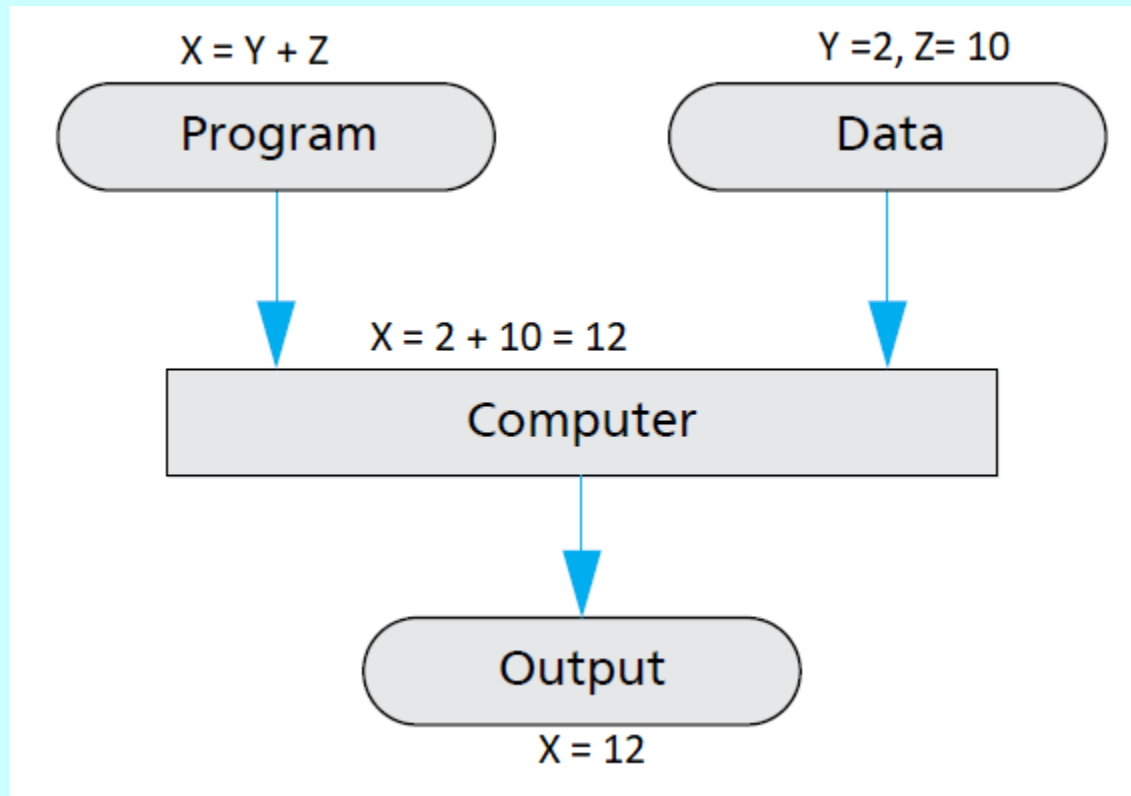
- The input to a computer can be thought of as consisting of two parts : a program and some data.
- The computer follows the instructions in the program, and in that way, performs some process.
- The data is what we conceptualize as the input to the program. For example, if the program adds two numbers, then the two numbers are the data.
- Whenever we give a computer both a program to follow and some data for the program, we are said to be running the program on the data, and the computer is said to execute the program on the data.

Running a Program



Example : Running a Program

Suppose you want a computer to execute a simple instruction $X = Y + Z$ to print the value of X on the output, where $Y = 2$ and $Z = 10$.



Analyzing Your First Program

```
# include <iostream>
using namespace std ;

int main()
{
    cout <<"Hello World"<< endl;
    return 0;
}
```

Output Terminal

Hello World

```
# include <iostream>
```

Tells the computer to include a service for input/output. You should simply remember to add this line into all programs that perform input and output.

```
using namespace std;
```

Tells the computer to use the “standard namespace”. Namespace is a mechanism for avoiding naming conflict in large programs. You do not need to be concerned about it. Simply add this statement in every program just below the #include directives.

```
int main()
{
    ...
    return 0;
}
```

The construction defines a *function* called `main` that “returns” an “integer” (that is, a whole number without a fractional part, called `int` in C++) with value 0. This value indicates that the program finished successfully. A **function** is a collection of programming instructions that carry out a particular task. Every C++ program must have a main function.

```
cout << "Hello World"<<endl;
```

To display values on the screen, you use an entity called `cout` and the `<<` operator (sometimes called the *insertion* operator).

You can send more than one item to `cout`. Use a `<<` before each one of them. For example,

```
cout << "The answer is: " << 6 * 7<<endl;
```

The `endl` symbol denotes an *end of line* marker. When this marker is sent to `cout`, the cursor is moved to the first column in the next screen row. If you don't use an end of line marker, then the next displayed item will simply follow the current string on the same line.

Modified First Program

```
# include <iostream>
using namespace std ;

int main()
{
    cout << "The answer is: " << 6 * 7<< endl;
    return 0;
}
```

Output Terminal

The answer is: 42

Modified First Program

```
#include <iostream>

using namespace std;

int main()
{
    cout << "The answer is: " << 6 * 7 << endl;
    return 0;
}
```

Output Terminal

The answer is: 42

Part 2: Variables and Assignments

Variables

- A variable is a storage location in a computer program.
- Each Variable has a name and holds a value.
- A variable can hold a number or data of other types. Initially we will confine our attention to variables that holds only numbers.

Number Type

- Numbers are two types: whole number and floating-point numbers
- To denote a whole number without a fractional part, we use integer `int` type.

Example:

```
4  
12
```

- When a fractional part is required, we use floating point numbers. The most used type for floating point numbers in C++ are called `float` and `double`.

Example:

```
4.32  
12.35
```

Variable Names: Identifiers

- Identifiers are used as names for variables in C++ program
- Variable names must start with a letter or underscore (`_`) character, the remaining characters must be letters, numbers, underscores.
- You can not use other symbols such as `?` Or `%`. Spaces are not permitted.

Correct:

```
x x1 x_1 sum RATE data2 EET_293 abc123
```

Incorrect:

```
12 3X %change data-1 myfirst.c EET-293
```

- Variable names are case sensitive. Radius and radius are different names.

Correct:

```
Radius radius
```

Correct:

```
rate RATE Rate
```

- You can not use reserved words such as `int`, `double`, `cout` as names

Incorrect:

```
cout int double
```


Assignment Statement

- An assignment statement place a new value into a variable.

Syntax:

```
variable = expression ;
```

Example:

```
sum = 6 ; → Assignment statement
```

- In the example, The assignment statement place the value 6 to the variable sum. An **expression** is a number, a variable, or a combination of numbers and variables and operation symbols.

Example
:

```
sum  = 0 ; → Assignment statement  
number = 5 ; → Assignment statement  
sum = sum + number ; → Assignment statement
```

Assignment Statement

- The = sign does not mean left hand side is equal to right hand side. Do not confuse assignment operation with the = used in algebra to denote the equality. The assignment operator is an instruction to place a value on the variable.

Example:

```
x = 0 ;    → Assignment statement  
x = x + 2 ; → Assignment statement
```

- It means to look up the value stored on x, and add 2 to it, and place the result back into x. The execution of $x=x+2$ increase the x by 2. In Mathematics, it would make no sense to write that $x=x+2$. No value can be equal itself plus 2.

Variable Declarations

- Every variable in C++ program must be declared. When you declare a variable, you are telling the computer that what kinds of data you will be storing in the variable.
- When you declare more than one variables, the variables are separated by comma.
- The kind of data a variable holds is called type, and the name for the type such as `int` or `double` is called type name.

Syntax:

```
Type_Name variable_Name1, variable_Name2, ...;
```

Example:

```
double distance;  
  
int count, num_of_dragons ;
```

Uninitialized Variables

- A variable has no meaningful value until a program gives it one. For example, if the variable `y` has not been given a value, then the following assignment statement is an error.
- A variable like `y` that has not been given any value is said to be **uninitialized**.

```
int y ;    → Variable declaration  
x = y + 2 ; → Assignment statement
```

Variable Declarations With initialization

- You can initialize a variable at the time that you declare a variable.

Syntax:

```
Type_Name variable_Name1= Expression_for_value_1,  
                variable_Name2= Expression_for_value_2,..;
```

Example:

```
double distance = 100.5;  
  
int count=0, num_of_dragons=5 ;
```

Part 3: INPUT AND OUTPUT

- a. cin statement
- b. cout statement

cout

- To display values of the variable as well as strings of text, you use an entity called `cout` and the `<<` operator (sometimes called the *insertion* operator).
- You can send more than one item to `cout`. Use a `<<` before each one of them.
- Notice that strings must be included in double quotes.

```
cout << "The area is:"<<square_area;
```

- The following statement tells computer to output two variables: the quoted string `"The area is:"` and the values of the variable `square_area`.

cout

- The below statement provides following output, assuming the value of the variable `square_area` is 25.

```
int square_area = 25;  
cout << "The area is:"<<square_area;
```

Output

The area is: 25

- New lines in output:** To start a new output line, you can include `\n` in a quoted string. `\n` tells the computer to start a new line in output.

```
int square_area = 25;  
cout << "The area is:\n"<<square_area;
```

Output

The area is:
25

Write a Program to print the area of a square, where length of each side is 5. The equation for area of the square $A = a^2$, where a is the length of each side

```
#include <iostream>

using namespace std;

int main()
{
    int a = 5, area;

    area = a*a;

    cout << "The area is:" << area<<endl;

    return 0;
}
```

Output
The area is:25

cin Statements

- A `cin` statement sets variable equal to values typed in at the keyboard. It is used to read user input.

Syntax:

```
cin >> Variable_1 >> Variable_2 >>.....;
```

Example 1:

```
cin>>price;
```

Example 2:

```
cin>>number>>size;
```

cin Statements

- When a program asks for user input, it should first print a message that tells the user which input is expected. Such a message is called prompt.

```
cout<<"Please enter the values:";
```

- `cin` object reads input from console window. You use extraction `>>` operator to place an input value into a variable.

```
cin>>price;
```

- When a program executes the input statement it waits for user to provide input. The user also needs to hit the `Enter` key so that the program accepts the input. After the user supplies the input, the number is placed on the value variable, and program continues.

Write a Program to ask user to enter the length of the side and print the area of a square. The equation for area of the square $A = a^2$, where a is the length of each side

```
#include <iostream>
using namespace std;
int main()
{
    // variable declaration
    int a, area ;
    cout<<"Enter the value of the side:";
    cin>>a;

    // area calculation
    area = a*a ;

    // printing area on the output terminal
    cout<<"The area is:"<<area;

    return 0;
}
```

Sample Output

Sample Output 1

Output
Enter the value of the side:3 The area is:9

Sample Output 2

Output
Enter the value of the side:5 The area is:25

Write a Program to print total price of an item . The sale price of the item is \$545.00, and the sales tax is 5%.

```
#include <iostream>
using namespace std;
int main()
{
    // Variable declaration
    double sale_price=545.00, sale_tax=0.05;
    double tax_amount, total_price;

    // total price calculation
    tax_amount= sale_price*sale_tax;
    total_price=sale_price+tax_amount;

    // Printing area on the output
    cout<<"the total price is: "<<total_price<<endl;

    return 0;
}
```

Output

the total price is:572.25

Part 4: Data Types And Expressions

- a. Arithmetic operators
- b. Expressions

Arithmetic Operators

- C++ supports four basic arithmetic operations: addition, subtraction, multiplication, and division- but it uses different symbols for multiplication and division.

Operation	C++ Symbol	Example (C++ Symbol)	Mathematics Symbol	Example (Mathematics Symbol)
Addition	+	$a + b$	+	$a + b$
Subtraction	-	$a - b$	-	$a - b$
Multiplication	*	$a * b$. or x	$a \cdot b$ or $a \times b$
Division	/	a / b	fraction bar or \div	$\frac{a}{b}$ or $a \div b$

Expressions

In C++, there are no symbols for powers and roots. To compute them, you must call *functions*. To take the square root of a number, you use the sqrt function. For example, \sqrt{x} is written as sqrt(x). To compute x^n , you write pow(x, n).

Mathematical Expression	C++ Functions	Comments
\sqrt{x}	sqrt (x)	Square root of x
x^n	pow (x, n)	Power of x
e^x	exp (x)	Exponent of x
x	abs (x)	Absolute value x
sinx	sin (x)	Sine of x
cosx	cos (x)	Cosine of x

Expressions

Mathematical Expression	C++ Expression
$b^2 - 4ac$	b*b - 4*a*c
$\frac{1}{x^2 + x + 3}$	1 / (x*x + x + 3)
$\frac{a + b}{c + d}$	(a+b) / (c+d)

Expressions

Mathematical Expression	C++ Expression
$\frac{x + y}{2}$	<code>(x + y) / 2</code>
$\frac{xy}{2}$	<code>x * y / 2</code>
$(1 + \frac{r}{100})^n$	<code>pow(1+r/100, n)</code>
$\sqrt{a^2 + b^2}$	<code>sqrt (a*a + b * b)</code>

To use the `sqrt` and `pow` functions, you must place the line `#include <cmath>` at the top of your program file.

Increment and Decrement

- The adding and subtracting of 1 with a variable is so common that there are special shorthand for it. The ++ operator adds 1 to a variable; the -- operator subtracts 1 from the variable.

```
int counter = 9;    // Variable declaration  
Counter++          // counter = counter + 1
```

The value on the counter becomes 10;

```
int counter = 9;    // Variable declaration  
Counter--          // counter = counter - 1
```

The value on the counter becomes 8;

Write a program to print the value of y, where $y = \sqrt{a^2 + b^2}$. The values of a and b are 10 and 5, respectively.

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    // Variable Declaration
    int a =10, b=5;
    double y;

    // Calculation of y
    y = sqrt(a*a+b*b);

    // printing the result
    cout << "The value of y is:" << y<<endl;
    return 0;
}
```

Output

The value of y is:11.1803

PROGRAM STYLE

Comments

- You should add comment which explains your code. This helps programmers who read your code understand your intent. Additionally, you will find comments helpful when you review your own programs.

```
int r = 5 ; // radius of the circle is 5 inch
```

- Compiler does not process comments at all. It ignores everything from // delimiter to the end of the line.

Naming Constants

- When a variable is defined with the reserved word `const`, its value can never change.
- Constants are generally written in capital letters to distinguish them visually from regular variables.

Syntax:

```
const Type_Name Variable_Name = Constant ;
```

Example:

```
const double PI = 3.14 ;
```


Write a Program to print the area of a circle, where the radius of the circle is 3.

The equation for area of the circle $A = \pi r^2$, where r is the radius

```
#include <iostream>
using namespace std;

int main()
{
    // Variable declaration
    int r=3;
    const double PI=3.14;
    double area;

    // Calculation of the area
    area = PI*r*r;

    // Printing the area
    cout<<"The area is: "<<area;

    return 0;
}
```

Output

The area is: 28.26

Part 5: FLOW OF CONTROL

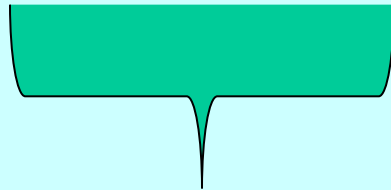
- a. if statement
- b. if-else statement
- c. while loop
- d. for loop

Decisions

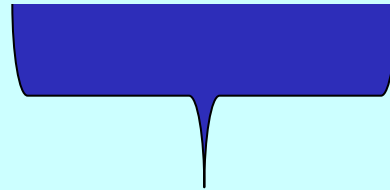
Sometimes, we need to decide based on a condition. For example,

If it rains today, I will stay home.

If it rains today, I will stay home.



Condition



decision

if Statement

- If statement is used to implement decision. When a condition is fulfilled, one set of statements is executed.

Syntax:

```
if (Boolean_Expression)
{
    statements
}
```

Example:

```
if (grade >= 60)
{
    cout<<" This student passed ";
}
```

Write a program to check a given grade. If the grade value is greater than or equal to 60, program prints “This student passed” on the console window.

```
#include <iostream>

using namespace std;

int main()
{
    int grade=82;

    if (grade >= 60)
    {
        cout<<" This student passed ";
    }

    return 0;
}
```

Output Terminal

This student passed

if –else Statement

- If statement is used to implement decision. When a condition is fulfilled, one set of statements is executed. Otherwise, another set of statements is executed.

Syntax:

```
if (Boolean_Expression)
    { statements1 }
else
    { statements2 }
```

Example:

```
if (grade >= 60)
{
    cout<<" This student passed ";
}
else
{
    cout<<" This student failed ";
}
```

Write a program to check a given grade. If the grade value is greater than or equal to 60, program prints “This student passed” on the console window. Otherwise, program prints “This student failed” on the console window

```
#include <iostream>

using namespace std;

int main()
{
    int grade=82;

    if (grade >= 60)
    {
        cout<<" This student passed ";
    }
    else
    {
        cout<<" This student failed ";
    }

    return 0;
}
```

Output Terminal

This student passed

Comparison operators

C++	Math Notations	Descriptions
>	>	Greater than
>=	\geq	Greater than or equal
<	<	Less than
<=	\leq	Less than or equal
==	=	Equal
!=	\neq	Not equal

while loop

- While loop executes block of code/statement repeatedly until a specific goal is attained. The repetition will continue as long as the condition remains true.

Syntax:

```
while (condition)
{
    statements
}
```

Example:

```
counter = 1;
while (counter <= 100)
{
    sum = sum + counter ;
    counter++ ;
}
```

while loop

- Using while loop, write a program to calculate the sum of first 100 Numbers. In other words, $1+2+3+4+5+6+\dots+100 = ?$
- Before calculating the sum for first 100 numbers, I want calculate the sum for first 3 numbers. In other words, $1+2+3 = ?$
- Once you understand the sum of first 3 numbers, then you will easily calculate the sum for first 100 numbers.
- The sum of the numbers ($1+2+3$) is 6, however, we want to do the summation with while loop.

Example:

A program to calculate summation of first 3 numbers?

$1+2+3 = ?$

while loop

Example:

A program to calculate summation of first 3 numbers?

$1+2+3= ?$

```
int sum = 0;
int counter = 1;
while (counter <= 3)
{
    sum = sum + counter ;
    counter++ ;
}

cout<<"The summation of 1+2+3 is:"<< sum<<endl;
```

While loop: summarized calculation

```
#include <iostream>

using namespace std;

int main()
{
    int sum = 0;
    int counter = 1;
    while (counter <= 3)
    {
        sum = sum + counter ;
        counter++ ;
    }

    cout<<"The summation of 1+2+3 is:"<< sum<<endl;

    return 0;
}
```

Output Terminal

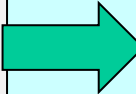
The summation of 1+2+3 is:6

sum	counter
0	1
1	2
3	3
6	4

While loop (Mystery inside): Do step by step calculation on your mind

1st iteration When,
counter is 1, sum is 0 at the beginning

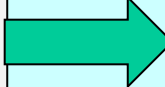
```
while (1<=3)
sum = sum + counter = 0+1 = 1 ;
counter = counter + 1= 1+1 = 2 ;
```



sum	counter
0	1
1	2

2nd iteration when, counter becomes 2

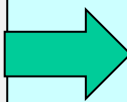
```
while (2<=3)
sum = sum + counter = 1+2 = 3 ;
counter = counter = 2+1 = 3 ;
```



sum	counter
3	3

3rd iteration when, counter becomes 3

```
while (3<=3)
sum = sum + counter = 3+3 = 6 ;
counter = counter = 3+1 = 4 ;
```



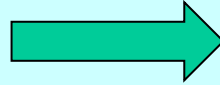
sum	counter
6	4

While loop (Mystery inside): Do step by step calculation on your mind

when counter becomes 4

```
while (4<=3) //Condition is false
// No action
// No increment
```

Final value of the
variables: (from the
3rd iteration)



sum	counter
6	4

Output Terminal

```
The summation of 1+2+3 is:6
```

while loop Example 1

What is the output of following program (when embedded in a complete program?)

```
int i = 0;
int sum = 0 ;
while (sum < 3)
{
    i++ ;
    sum = sum + i ;
    cout<<i<<" " <<sum<<endl;
}
```

Output Terminal

```
1 1
2 3
```

After 1st iteration

After 2nd iteration

i	sum
0	0
1	1
2	3

while loop Example 2

What is the output of following program (when embedded in a complete program?)

```
i = 0;
sum = 0 ;
while (sum < 0)
{
    i++ ;
    sum = sum + i ;
    cout<<i<<" "<<sum;
}
```

No output

Output Terminal

i	sum
0	0

The statement `sum < 0` is false when the condition is first checked, and the loop is never executed.

for loop

- Like While loop, for loop executes block of code/statement repeatedly until a specific goal is attained. The repetition will continue as long as the condition remains true.
- The for loop is used when a value runs from a starting point to an ending point with a constant increment or decrement. for loop is count controlled.
- for loop is count controlled as it executes repeatedly until the counter based condition remains true.

for loop

- The initialization is executed once, before the loop is entered.
- The condition is checked before each iteration
- The update is executed after each iteration.

Syntax:

```
for (initialization; condition; update)
{
    statements
}
```

Example:

```
int counter;
int fact = 1;
for ( counter = 1; counter <= 3; counter++)
{
    fact = fact*counter ;
}
```

for loop Example

- Using `for` loop, write a program to calculate the factorial of 10. In other words, $1.2.3.4.5.6.7.8.9.10 = ?$
- Before calculating factorial of 10, I want calculate factorial of 3. In other words, $1.2.3 = ?$
- Once you understand the factorial of 3, then you will be able to easily calculate the factorial of 3..
- The factorial of 3 is $= 1.2.3 = 6$, however, we want to do the calculation using `for` loop.

Example:

A program (using `for` loop) to calculate the factorial of 3 ?

$1.2.3 = ?$

Using for loop, write a program to calculate the factorial of 3.

```
#include <iostream>
using namespace std;
int main()
{
    int counter;
    int fact = 1;
    for ( counter = 1; counter <= 3; counter++)
    {
        fact = fact*counter ;
    }
    cout<<"The factorial of 3 is: "<< fact;

    return 0;
}
```

Output Terminal
The factorial of 3 is:6

for loop (Mystery inside): Do step by step calculation on your mind

1st iteration When,
counter is 1, fact is 1 at the beginning

check
condition → (1 ≤ 3)
action → fact = fact * counter = 1 * 1 = 1 ;
increment → counter = counter + 1 = 1 + 1 = 2 ;

fact	counter
1	1
1	2

2nd iteration when, counter becomes 2

check
condition → (2 ≤ 3)
action → fact = fact * counter = 1 * 2 = 2
increment → counter = counter + 1 = 2 + 1 = 3 ;

fact	counter
2	3

3rd iteration when, counter becomes 3

check
condition → (3 ≤ 3)
action → fact = fact * counter = 2 * 3 = 6 ;
increment → counter = counter + 1 = 3 + 1 = 4 ;

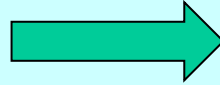
fact	counter
6	4

for loop (Mystery inside): Do step by step calculation on your mind

when counter becomes 4

check	→	(4<=3) //Condition is false
condition	→	// No action
action	→	// No increment
increment	→	

Final value of the
variables: (from the
3rd iteration)



fact	counter
6	4

Output Terminal

The factorial of 3 is:6

for loop Example 1

What is the output of following program (when embedded in a complete program?)

```
for (int count= 1; count < 5; count++)  
{  
    cout<<(2*count) << " ";  
}
```

Output Terminal
2 4 6 8

for loop Example 2

What is the output of following program (when embedded in a complete program?)

```
for (int n = 6; n > 0; n=n-2)
{
    cout<<" Hello " ;
    cout<<n<<endl ;
}
```

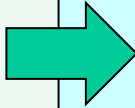
Output Terminal

```
Hello 6
Hello 4
Hello 2
```


for loop Example 3 : Count iterations

What is the output of following program (when embedded in a complete program?)

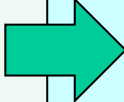
```
for ( int i = 1; i <= 5; i++)  
{  
    cout<<i<<endl;  
}
```



Output

1
2
3
4
5

```
for ( int i = 1; i < 5; i++)  
{  
    cout<<i<<endl;  
}
```



Output

1
2
3
4

Common Loop Algorithms

Sum and Average Value

- Computing the sum of a number of inputs is a very common task. Keep a running `sum`, a variable to which you add each input value. The `sum` should be initialized with 0.
- Note that the `sum` variable is declared outside the loop. We want the loop to update a single variable

A Program to calculate the Sum and Average from 5 integer numbers entered by User.

```
#include <iostream>
using namespace std;

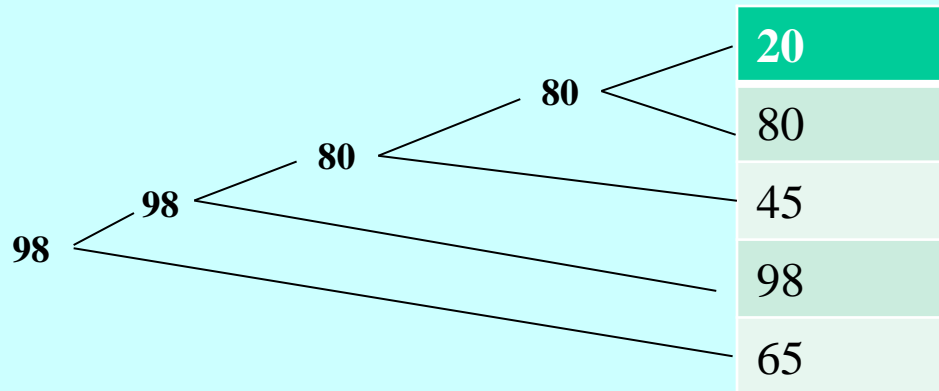
int main()
{
    int i, n=5, input, sum = 0, average;
    cout<<"Enter the integer numbers:";
    for (i=1; i<=n; i++)
    {
        cin>>input;
        sum = sum + input;
    }
    average = sum / n;
    cout<<"Sum: "<<sum<<" Average: "<<average;
}
```

Output

Enter sequence of integer values:
20 30 40 50 60.
Sum: 200 Average: 40

Finding Largest Number

- To compute the largest value in a sequence, keep a variable that stores the largest element that you have encountered, and update it when you find a larger one.
- Suppose, you want to find the largest of the following numbers: 20 80 55 98 65. Initially, you need to consider first element (20) as the largest, and store it to a variable. Now this variable will be compared with next element (80), this process will continue on the following manner.



A program to find largest number from 4 integer values entered by the user.

```
#include <iostream>
using namespace std;
int main ()
{
    int large,input, n=4;
    cout<<"Enter the input values: ";
    cin>>large;
    for (int i=1; i<=n-1; i++)
    {
        cin>>input;
        if (input>large)
        {
            large=input;
        }
    }
    cout<<"largest number is: "<<large << endl;
}
```

Output

Enter the input values: 23 78 90 56
largest number is: 90

Note: The figures, text etc included in slides are borrowed from books, other sources for academic purpose only. The author does not claim any originality.

Source:

- **Problem Solving with C++ (10th Edition) by Walter Savitch**
- **C++ for Everyone 2nd Edition by Cay S. Horstmann**