ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Data Structures and Algorithms

Lecture 2

© Fall 2025- **Dr. Hesham Sakr**
Building, office No. 12
Hesham.sakr@sut.edu.eg

INTRODUCTION TO COMPUTER PROGRAMMING

WHAT IS A COMPUTER?

- Computer
  - Device capable of performing computations and making logical decisions
  - Computers process data under the control of sets of instructions called computer programs
- Hardware
  - Various devices comprising a computer
  - Keyboard, screen, mouse, disks, memory, CD-ROM, and processing units
- Software
  - Programs that run on a computer

WHAT IS PROGRAMMING?

Programming is taking

A *problem*

Find the area of a rectangle

A set of *data*

length

width

A set of *functions*

area = length * width

Then,

Applying functions to data to solve the problem

PROGRAMMING – WHY?

- *Computers are used for many different purposes in many different situations.*
- *A program is a set of instructions that tell a computer what to do.*
- *A computer cannot do anything unless it has a program to tell it what to do.*

SOFTWARE CATEGORIES

- Programs can also be called software.
  - Software refers to the computer programs that a computer uses to complete a task.
- System SW
  - Programs written for computer systems
    - Compilers, operating systems, …
- Application SW
  - Programs written for computer users
    - Word-processors, spreadsheets, & other application packages

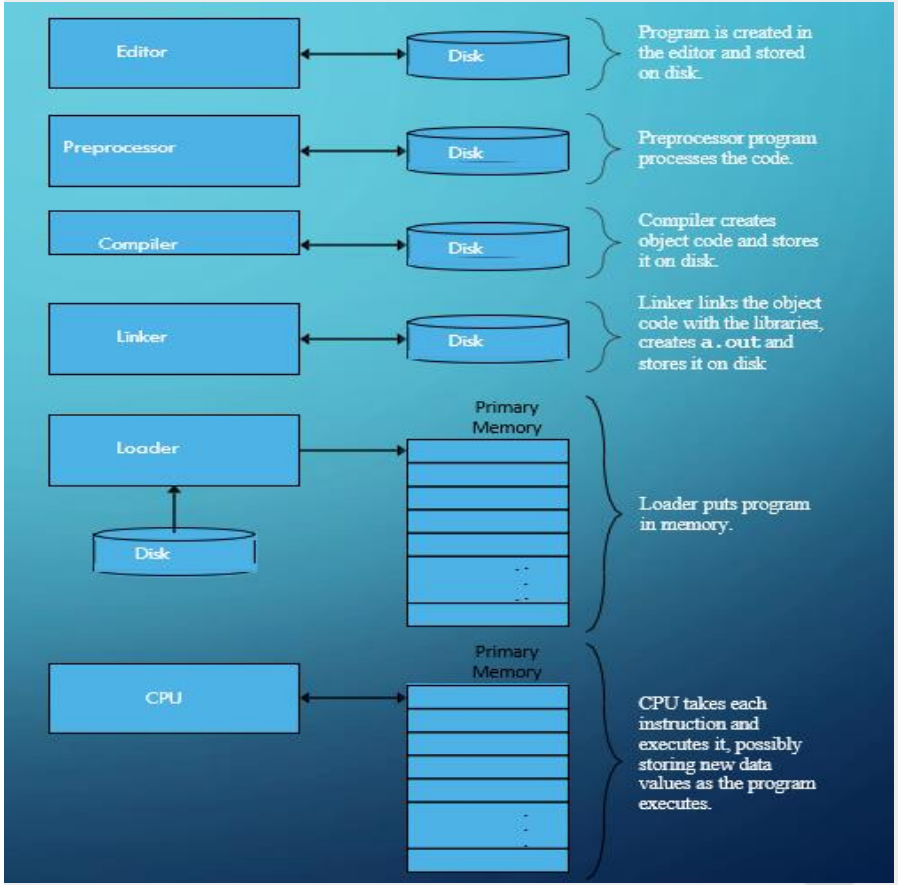
PROGRAMMING LANGUAGES

- Many different programming languages
  - Java, C, **C++**, Scheme, Haskell, Visual Basic, Perl, Python, Tcl/Tk, Pascal, Basic, Lisp, Prolog, Cobol, C#, Smalltalk, Eiffel, Fortran, Ada, Mathematica, MatLab

BASICS OF C++

BASICS OF A TYPICAL C++ ENVIRONMENT

- Phases of C++ Programs:
  1. *Edit*
  2. *Preprocess*
  3. *Compile*
  4. *Link*
  5. *Load*
  6. *Execute*



A SIMPLE PROGRAM

```
// Simple program.cpp

// A first program in C++

#include <iostream>

   int main()

{

      std::cout << "Welcome to C++!\n";

      return 0;       // indicate that program ended successfully

   }
```

```
Welcome to C++!
```

COMMENTS

- /* You can use this comment style, */
- /* but you must be // very careful about mixing them */
- // It's best to use this style for 1 line or partial lines
- /* And use this style when your comment
- consists of multiple lines */

PRINTING A LINE OF TEXT

- **`std::cout`**
  - standard output stream object
  - “connected” to the screen
  - we need the **`std::`** to specify what "namespace" **`cout`** belongs to
    - we shall remove this prefix with **`using`** statements
- **`<<`**
  - stream insertion operator
  - value to the right of the operator (right operand) inserted into output stream (which is connected to the screen)

```
std::cout << " Welcome to C++!\n"
```

PRINTING A LINE OF TEXT (II)

There are multiple ways to print text

Escape Sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
\\	Backslash. Used to print a backslash character.
\"	Double quote. Used to print a double quote character.

PRINTING A LINE WITH MULTIPLE STATEMENTS

```
   // Printing a line with multiple statements
#include <iostream>
       int main()
       {
      std::cout << "Welcome ";
       std::cout << "to C++!\n";
 return 0;   // indicate that program ended successfully
}
```

```
Welcome to C++!
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

BASICS OF C++ CONTINUE

VARIABLES

- location in memory where a value can be stored for use by a program
- must be declared with a name and a data type before they can be used
- Some common data types are:
- int- integer numbers
- char - characters
- double - floating point numbers
- Example: int x;
- Declares a variable named xof type int
- Example: int x, number1;
- Declares two variables, each of type int

VARIABLES (CONTINUE)

- >> (stream extraction operator)
- When used with std::cin, waits for user to input a value and stores the value in the
- variable to the right of the operator.
- Example:
- int x;
- std::cin >> x;
- waits for user input, then stores input in x
- • = (assignment operator)
- assigns value to a variable
- binary operator (has two operands)
- sum = number1 + number2;

ADDING TWO VARIABLES

```
// Addition program
#include <iostream>

   int main()
     {
       int number1, number2, sum;            // declaration

       std::cout << "Enter first integer\n";   // prompt
       std::cin >> number1;                    // read an integer
       std::cout << "Enter second integer\n"; // prompt
       std::cin >> number2;                    // read an integer
       sum = number1 + number2;               // assignment of sum
       std::cout << "Sum is " << sum << std::endl; // print sum

       return 0;    // indicate that program ended successfully
   }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

ARITHMETIC CALCULATIONS

- Arithmetic calculations are used in most programs
- special notes:
- use *for multiplication and /for division
- integer division truncates remainder
- 7 / 5evaluates to 1
- modulus operator returns the remainder
- 7 % 5evaluates to 2

- Operator precedence
- some arithmetic operators act before others (i.e., multiplication before addition)
- be sure to use parenthesis when needed
- Example: Find the average of three variables a, band c
- do not use: a + b + c / 3
- use: (a + b + c) / 3

ARITHMETIC CALCULATIONS (II)

- Arithmetic operators:

C++ operation	Arithmetic operator	Algebraic expression	C++ expression
Addition	+	f + 7	f + 7
Subtraction	-	p – c	p - c
Multiplication	*	bm	b * m
Division	/	x / y	x / y
Modulus	%	r mod s	r % s

- Rules of operator precedence:

Operator(s)	Operation(s)	Order of evaluation (precedence)
()	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses “on the same level” (i.e., not nested), they are evaluated left to right.
*, /, or %	Multiplication Division Modulus	Evaluated second. If there are several, they re evaluated left to right.
+or -	Addition Subtractio n	Evaluated last. If there are several, they are evaluated left to right.

EXAMPLE : CALCULATE THE PRODUCT OF THREE INTEGERS

```
#include <iostream> // allows program to perform input and output using namespace std; // program uses names from the std namespace int main()
{
int x; // first integer to multiply
int y; // second integer to multiply int z; // third integer to multiply
int result; // the product of the three integers
cout << "Enter three integers: "; // prompt user for data  cin >> x >> y >> z; // read three integers from user  result = x * y * z; // multiply the three integers; store result
cout << "The product is " << result << endl; // print result; end line
} // end function main
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

CONTROL STATEMENTS

IF STATEMENT

- `if` structure - decision based on truth or falsity of condition. If condition met execute,

- If student's marks are greater than or equal to 60 *Print* "Passed"

- if (marks>= 60)
- std: : cout << "Passed";

A SIMPLE PROGRAM USING IF STATEMENT

```
#include <iostream>

   int main()
     {
      int marks;            // declaration

      std::cout << "Enter student marks\n";   // prompt
      std::cin >> marks;                     // read students marks

        if ( marks  >= 60 )
        std::cout << "Passed";

      return 0;    // indicate that program ended successfully

   }
```

```
Enter student marks
90
Passed
```

EQUALITY AND RELATIONAL OPERATORS:

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	**>**	**x > y**	**x** is greater than **y**
<	**<**	**x < y**	**x** is less than **y**
	>=	**x >= y**	**x** is greater than or equal to **y**
	<=	**x <= y**	**x** is less than or equal to **y**
Equality operators			
=	**==**	**x == y**	**x** is equal to **y**
	!=	**x != y**	**x** is not equal to **y**

USINGSTATEMENTS

- eliminate the need to use the std::prefix
- allow us to write coutinstead of std::cout
- at the top of the program write using std::cout; using std::cin; using std::endl;
- to use those functions without writing std::

```
#include <iostream>
using std::cout;          // program uses
cout using std::cin;      // program uses
cin
using std::endl;          // program uses endl


int main()

 { int num1, num2;
    cout << "Enter two integers";

    cin >> num1 >> num2;            // read two integers

    if ( num1 == num2 )
        cout << num1 << " is equal to " << num2 << endl;

    if ( num1 != num2 )
        cout << num1 << " is not equal to " << num2 << endl;

    if ( num1 < num2 )
        cout << num1 << " is less than " << num2 << endl;

    if ( num1 > num2 )

        cout << num1 << " is greater than " << num2 << endl; return 0;
                        // indicate that program ended successfully
  }
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

ELSE IF STATEMENT

ELSE IF STATEMENT

- `if` structure - decision based on truth or falsity of condition. If condition met execute,

If student's marks are greater than or equal to 60
Print "Passed" else print " Failed"

```
if ( marks  >= 60 )
std: : cout << "Passed";
else
std: : cout << "Failed";
```

```
if ( marks  >= 60 )
std: : cout << "Passed";
else if ( marks  < 60 )
std: : cout << "Failed";
```

EXAMPLE (1)

```
#include <iostream>
using std::cout;                    // program uses cout
using std::cin;                     // program uses cin
int main()
{
int time;                           // declaration
cout << "Enter time \n";      // prompt
cin >> time;                        // read time

if (time < 10)

cout << "Good morning.";

else if (time < 20)

cout << "Good day.";

else

  cout << "Good evening.";

return 0;   }
```

EXAMPLE (2)

```
#include <iostream>

using std::cout;  // program uses cout

using std::endl;  // program uses cout

int main () {

   // local variable declaration:

   int a = 100;

      if( a == 10 )

      cout << "Value of a is 10" << endl;

     else if( a == 20 )

      cout << "Value of a is 20" << endl;

     else if( a == 30 )

      cout << "Value of a is 30" << endl;

      else

      cout << "Value of a is not matching" << endl;

   return 0;  }
```

EXAMPLE (3)

If student's grade is greater than or equal to 90

Print "A"

Else

If student's grade is greater than or equal to 80

Print "B"

Else

If student's grade is greater than or equal to 70

Print "C"

Else

If student's grade is greater than or equal to 60

Print "D"

Else

Print "F"

```
#include <iostream>
using std::cout;                        // program uses cout
using std::cin;                         // program uses cin
int main()
{
int marks;                              // declaration
cout << "Enter student marks\n";        // prompt
cin >> marks;                           // read students marks
if (marks >= 90 )                       // 90 and above gets "A"
cout << "A";
else if (marks >= 80 )                        // 80-89 gets "B"
cout << "B";
else if (marks >= 70 )                        // 70-79 gets "C"
cout << "C";
else if (marks >= 60 )                       // 60-69 gets "D"
cout << "D";
else                                    // less than 60 gets "F"
cout << "F";
return 0;}
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

ASSIGNMENT AND SWITCH STATEMENT

Assignment

- Q1) Write a C++ program to show that: If integer variable opCode has the value 1, read in double values for X and Y and calculate and print their sum.
- #include <iostream> int main()
- {
- int opCode;
- std::cout << "Enter opCodes\n"; std::cin >> opCode;
- if (opCode == 1)
- {
- double X , Y , SUM; std::cout << "Enter X\n"; std::cin >> X;
- std::cout << "Enter Y\n"; std::cin >> Y;
- SUM = X+Y ;
- std::cout << " SUM = " << SUM ;
- }
- return 0; // indicate that program ended successfully
- }

Write a C++ program to Assign a value to an integer variable cost depending on the value of integer variable distance as follows:

Distance	Cost
0 through 100	5
More than 100 but not more than 500	8
More than 500 but less than 1000	10
1000 or more	12

```
#include <iostream>
int main()
{
int Distance , Cost;
 std::cout << "Enter Distance\n";
 std::cin >> Distance;
 if ( 0 < Distance && Distance <= 100 ) Cost = 5;
 if ( 100 < Distance  && Distance <= 500 )  Cost = 8;
 if ( 500 < Distance  &&  Distance < 1000 ) Cost = 10;
 if ( 1000 <= Distance ) Cost = 12;
 std::cout << " Cost = " << Cost ;
return 0;
}
```

SWITCH STATEMENT

- switch(expression) {
- case x:
- // code block break;
- case y:
- // code block break; default:
- // code block
- }

EXAMPLE (1)

```
#include <iostream>
using namespace std;
int main() {
  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;
  }
  return 0;
}
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

SWITCH AND FOR STATEMENTS

SWITCH STATEMENT (CONTINUE) EXAMPLE 1

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

int main() { int x
= 2;
    switch (x)
    {
        case 1:
            cout << "Choice is 1";
            break;
        case 2:
            cout << "Choice is 2";
            break;
        case 3:
            cout << "Choice is 3";
            break;
        default:
            cout << "Choice other than 1, 2 and 3";
            break;
    }
return 0;
}
```

EXAMPLE 2

```
#include  <iostream>
using namespace std;
int main () {
   // local variable declaration:
   char grade = 'D';
   switch(grade) {
      case 'A' :
         cout << "Excellent!" << endl;
         break;
      case 'B' :
      case 'C' :
         cout << "Well done" << endl;
         break;
      case 'D' :
         cout << "You passed" << endl;
         break;
      case 'F' :
         cout << "Better try again" << endl;
         break;
      default :
         cout << "Invalid grade" << endl;
                        }
   cout << "Your grade is " << grade << endl;
   return 0;    }
```

C++ FOR LOOP

```
for (statement 1; statement 2; statement 3)
 {
  // code block to be executed
 }
```

EXAMPLE 1: PRINTING NUMBERS FROM 1 TO 5

```
#include <iostream>
using namespace std;
int main() {
        for (int i = 1; i <= 5; i++) {
        cout << i << " ";
              }
    return 0;
}
```

Output

1 2 3 4 5

EXAMPLE 2: PRINTING NUMBERS FROM 10 TO 19

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

int main () {
   // for loop execution
   for( int a = 10; a < 20; a = a + 1 ) {
      cout << "value of a: " << a << endl;
   }
   return 0;   }
```

Output

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

EXAMPLE 3: DISPLAY A TEXT 5 TIMES

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 5; i++) {
        cout <<  "Hello World! " << endl;
    }
    return 0;
}
```

Output

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

EXAMPLE 4: FIND THE SUM OF FIRST N NATURAL NUMBERS

```
// C++ program to find the sum of first n natural numbers
// positive integers such as 1,2,3,...n are known as natural numbers
#include <iostream>
using namespace std;
int main() {
    int num, sum;
    sum = 0;
    cout << "Enter a positive integer: ";
    cin >> num;
    for (int i = 1; i <= num; i++) {
        sum = sum + i;
    }
    cout << "Sum = " << sum << endl;
    return 0;
}
```

Output

```
Enter a positive integer: 10
Sum = 55
```

EXAMPLE 5: FIND THE FACTORIAL OF A NUMBER

```
#include <iostream>
using namespace std;
int main()
{
        int x, num, factorial = 1;
        cout << "Type positive number: ";
        cin >> num;
        for (x = 1; x <= num; ++x) {
                factorial *= x;   // factorial = factorial * x;
        }
        cout << "Factorial of " << num << " = " << factorial;
        return 0;
}
```

Output

```
Type positive number: 4
Factorial of 4 = 24
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

NESTED FOR

// C++ PROGRAM TO DISPLAY A PATTERN WITH 5 ROWS AND 3 COLUMNS

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

int main() {

   for (int i = 1; i <= 5; ++i) {
      for (int j = 1; j <= 3; ++j) {
         cout << " *  ";
      }
      cout << endl;
   }
  return 0;
}
```

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

// C++ PROGRAM TO DISPLAY 7 DAYS OF 3 WEEKS

```
#include  <iostream>
using namespace std;
int main() {
   for (int i = 1; i <= 3 ; ++i) {
      cout << "Week: " << i << endl;

      for (int j = 1; j <= 7 ; ++j) {
         cout << "    Day:" << j << endl;

   }
   return 0;
```

```
Week: 1
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
Week: 2
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
Week: 2
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
```

EXAMPLE: BREAK INSIDE NESTED LOOPS

```
#include  <iostream>
using namespace std;
int main() {
   for (int i = 1; i <= 3 ; ++i) {

        if (i == 2) {
            break;
        }
      cout << "Week: " << i << endl;

      for (int j = 1; j <= 7 ; ++j) {

         cout << "    Day:" << j << endl;
```

```
Week: 1
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
```

EXAMPLE: CONTINUE INSIDE NESTED LOOPS

```
#include  <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 3 ; ++i) {

        if (i == 2) {
            continue;
        }
       cout << "Week: " << i << endl;

       for (int j = 1; j <= 7 ; ++j) {

          cout << "    Day:" << j << endl;
```

```
Week: 1
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
Week: 3
    Day:1
    Day:2
    Day:3
    Day:4
    Day:5
    Day:6
    Day:7
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

WHILE AND DO- WHILE STATEMENTS

WHILE STATEMENT

The syntax of a while loop in C++

```
while(condition) {
statement(s);
}
```

EXAMPLE (1)

```
// C++ Program to print numbers from 1 to 5 using while loop

#include <iostream>
using namespace std;

int main () {
   // Local variable declaration:
   int i = 1;

   // while loop execution
   while( i < 6 ) {
      cout << i << " ";
      i++;
   }

   return 0;
}
```

```
1 2 3 4 5
```

EXAMPLE (2)

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

int main () {
   // Local variable declaration:
   int a = 10;

   // while loop execution
   while( a < 20 ) {
      cout << "value of a: " << a << endl;
      a++;
   }
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

EXAMPLE (3)

```
// program to find the sum of positive numbers
// if the user enters a negative number, the loop ends
// the negative number entered is not added to the sum

#include <iostream>
using namespace std;
int main() {
    int number;
    int sum = 0;
    // take input from the user
    cout << "Enter a number: ";
    cin >> number;
    while (number >= 0) {
        // add all positive numbers
        sum = sum + number;
        // take input again if the number is positive
        cout << "Enter a number: ";
        cin >> number;
    }
cout << "\nThe sum is " << sum << endl;
        return 0;
}
```

```
Enter a number: 6
Enter a number: 12
Enter a number: 7
Enter a number: 0
Enter a number: -2

The sum is 25
```

DO WHILE STATEMENT

The syntax of a do while loop in C++

```
do {
    // body of loop;
}
while (condition);
```

EXAMPLE (1)

```
// C++ Program to print numbers from 1 to 5 using do…while loop

#include <iostream>

using namespace std;

int main() {
    int i = 1;

    // do...while loop from 1 to 5
    do {
        cout << i << " ";
         i ++;
    }
    while (i <= 5);

    return 0;
}
```

```
1 2 3 4 5
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

C++ ARRAYS

C++ ARRAYS

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type, specify the name of the array followed by **square brackets** and specify the number of elements it should store:

Declaring a variable that holds an array of four strings

```
string cars[4];
```

To give values to the array

```
string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
```

CHANGE AN ARRAY ELEMENT

To change the value of a specific element, refer to the index number:

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

int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  cars[0] = "Opel";
  cout << cars[0];
  return 0;
}
```

```
Opel
```

C++ ARRAYS AND LOOPS

The following example outputs all elements in the **cars** array using for loop:

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

int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  for(int i = 0; i < 4; i++) {
    cout << cars[i] << "\n";
  }
  return 0;
}
```

```
Volvo
BMW
Ford
Mazda
```

- The following example outputs the index of each element together with its value:

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

int main() {
  string cars[4] = {"Volvo", "BMW", "Ford", "Mazda"};
  for(int i = 0; i < 4; i++) {
    cout << i << ": " << cars[i] << "\n";
  }
  return 0;
}
```

```
0: Volvo
1: BMW
2: Ford
3: Mazda
```

EXAMPLE: TAKE INPUTS FROM USER AND STORE THEM IN AN ARRAY

```
#include <iostream>
using namespace std;
int main() {
    int numbers[5];
    cout << "Enter 5 numbers: " << endl;
    //  store input from user to array
    for (int i = 0; i < 5; ++i) {
        cin >> numbers[i];
    }
    cout << "The numbers are: ";
    //  print array elements
    for (int n = 0; n < 5; ++n) {
        cout << numbers[n] << "  ";
    }
    return 0;
}
```

```
Enter 5 numbers:
11
12
13
14
15
The numbers are:    12  13      15
11                  14
```

EXAMPLE: DISPLAY SUM AND AVERAGE OF ARRAY ELEMENTS USING FOR LOOP

```
#include <iostream>
using namespace std;
int main() {
     double numbers[] = {7, 5, 6, 12, 35, 27};
    double sum = 0;
    double average;
    for (int i = 0; i < 6; i++) {


    }
   average = (sum/6);
    cout << "\nThe Sum = " << sum << endl;
    cout << "\nThe average = " << average << endl;
        return 0;
```

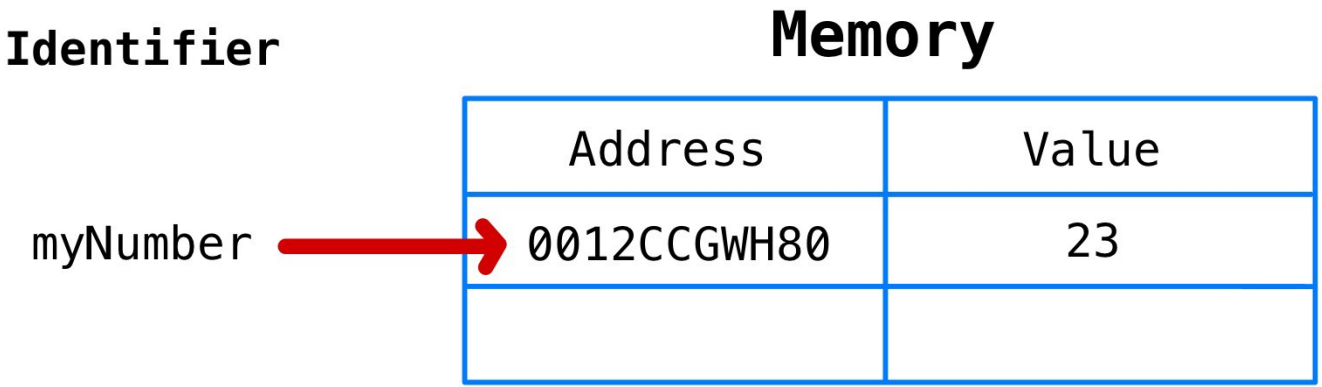
```
The Sum = 92

The average = 15.3333
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

C++ REFERENCES AND POINTERS

A VARIABLE IN A MEMORY



C++ REFERENCES

A reference variable is a "reference" to an existing variable, and it is created with the & operator:

```
string food = "Pizza";  // food variable
string &meal = food;    // reference to food
```

Now, we can use either the variable name food or the reference name meal to refer to the food variable:

```
int main() {
  string food = "Pizza";
  string &meal = food;

  cout << food << "\n";
  cout << meal << "\n";
  return 0;
}
```

```
Pizza
Pizza
```

MEMORY ADDRESS

When a variable is created in C++, a memory address is assigned to the variable. And when we assign a value to the variable, it is stored in this memory address.

To access it, use the & operator, and the result will represent where the variable is stored:

```
string food = "Pizza";
cout << &food; // Outputs 0x6dfed4
```

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

int main() {
  string food = "Pizza";

  cout << &food;
  return 0;
}
```

C++ POINTERS

A **pointer** is a variable that **stores the memory address as its value**

```
string food = "Pizza";  // A string variable
string* ptr = &food;  // A pointer variable that stores the address of food
```

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string food = "Pizza";  // A string variable
  string* ptr = &food;  // A pointer variable that stores the
address of food
  // Output the value of food
  cout << food << "\n";
  // Output the memory address of food
  cout << &food << "\n";
  // Output the memory address of food with the pointer
  cout << ptr << "\n";
  return 0;
}
```

GET MEMORY ADDRESS AND VALUE

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

int main() {
  string food = "Pizza";  // Variable declaration
  string* ptr = &food;    // Pointer declaration

  // Reference: Output the memory address of food with the
pointer
  cout << ptr << "\n";

  // Dereference: Output the value of food with the pointer
  cout << *ptr << "\n";
  return 0;
}
```

```
0x6dfed4
Pizza
```

AN EXAMPLE FOR POINTERS

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

int main ()
{
  int firstvalue, secondvalue;
  int * mypointer1;
  int * mypointer2;

  mypointer1 = &firstvalue;
  *mypointer1 = 10;
  mypointer2 = &secondvalue;
  *mypointer2 = 20;
  cout << "firstvalue is " << firstvalue << '\n';
  cout << "The address of the firstvalue is " << mypointer1 << '\n';
  cout << "secondvalue is " << secondvalue << '\n';
  cout << "The address of the secondvalue is " << mypointer2 << '\n';
  return 0;
}
```

```
firstvalue is 10
The address of the firstvalue is 0x7ffed6077248
secondvalue is 20
The address of the secondvalue is 0x7ffed607724c
```

EXAMPLE: MODIFY THE POINTER VALUE

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string food = "Pizza";
string* ptr = &food;
cout << food << "\n";
cout << &food << "\n";
cout << *ptr << "\n";
    // Change the value of the pointer
  *ptr = "rice";
    // Output the new value of the pointer
  cout << *ptr << "\n";
    // Output the new value of the food variable
  cout << food << "\n";
  return 0;
}
```

ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Thank You