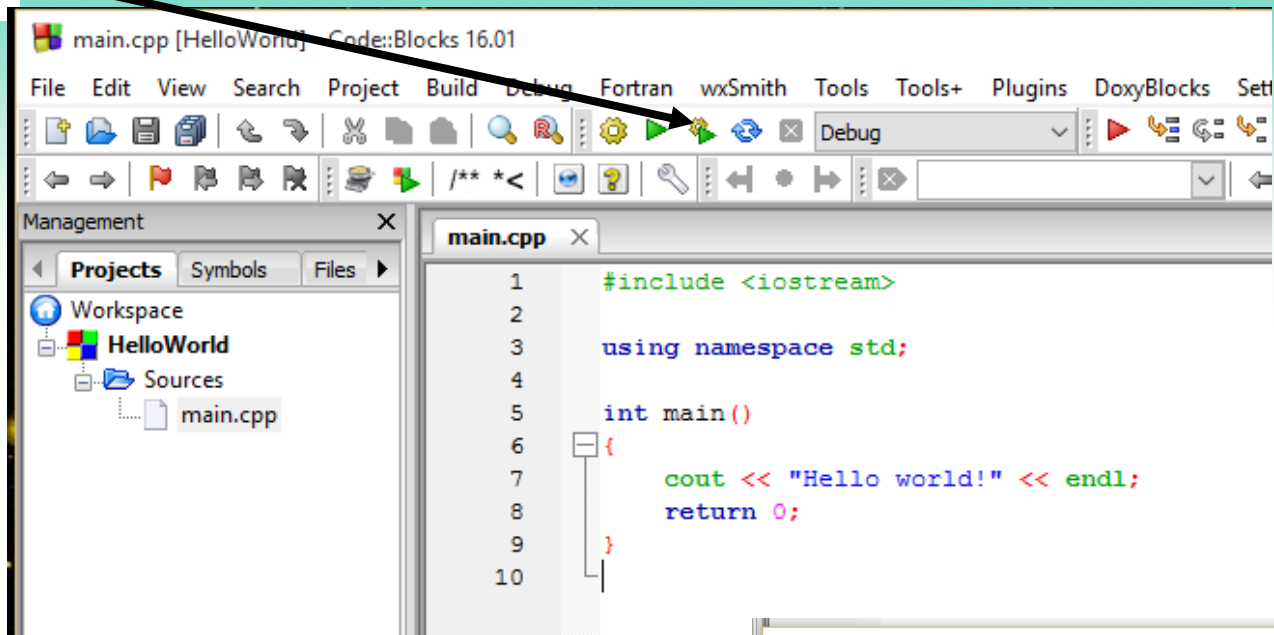


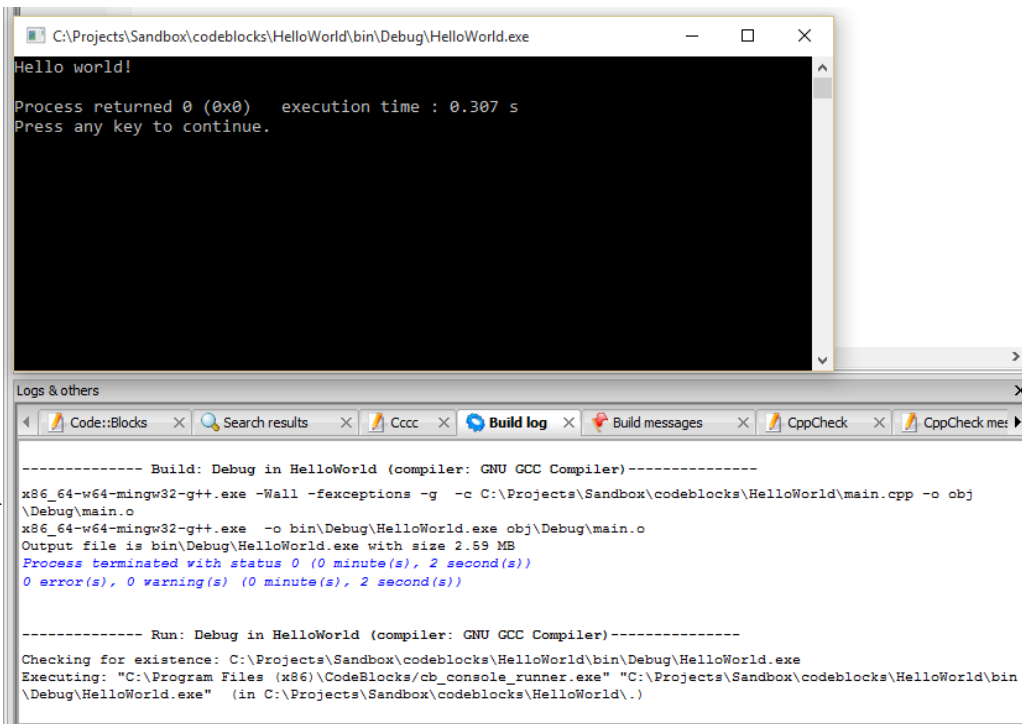
Basic C++

Day 1



Hello, World!

Console window:



Build and compile

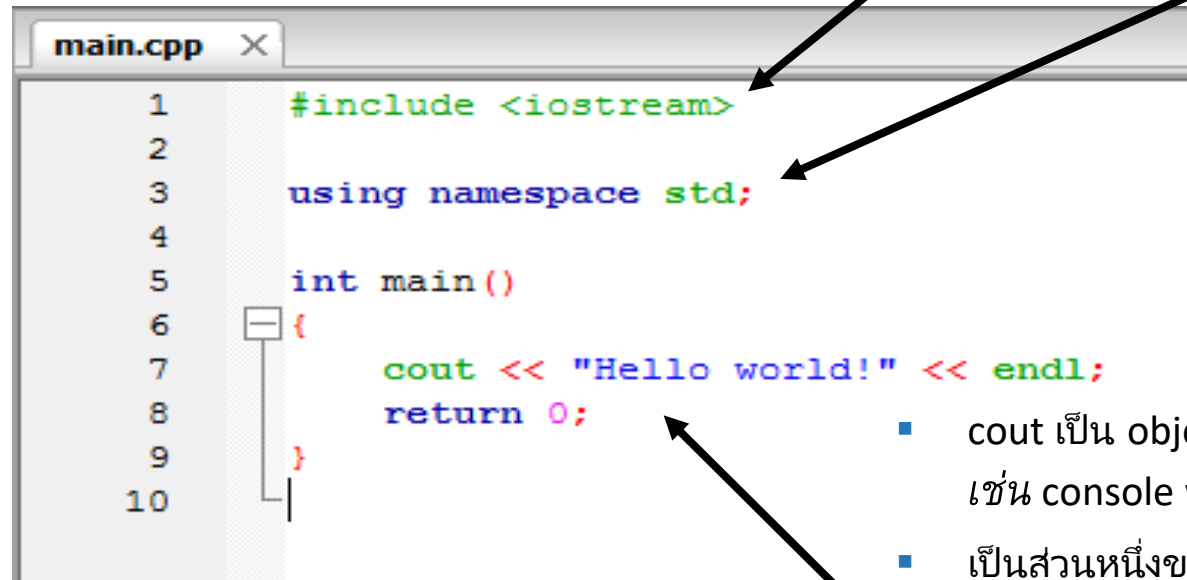
Hello, World! explained

```
main.cpp X
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

main – เป็นจุดเริ่มต้นของทุกๆ โปรแกรม
ใน C++ มันจะทำการ returns integer
value ให้กับระบบปฏิบัติการ และ (ในกรณี
นี้) ไม่รับ arguments: `main()`

คำสั่ง `return` จะส่งค่าจำนวนเต็มกลับไปยังระบบปฏิบัติการ
หลังจากทำงานเสร็จสิ้น ค่า 0 หมายถึง 'ไม่มีข้อผิดพลาด'
โปรแกรม C++ จะต้องส่งค่าจำนวนเต็มกลับ.

Hello, World! explained



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello world!" << endl;
8      return 0;
9  }
10
```

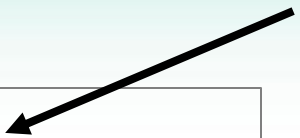
loads *header* file
containing function and
class definitions

โหลด *namespace* ชื่อ *std*
โดย *namespace* ใช้เพื่อ
ความสะดวกของ
โปรแกรมเมอร์

- *cout* เป็น object ที่เขียนข้อความไปยังอุปกรณ์ เช่น console window.
- เป็นส่วนหนึ่งของไลบรารีมาตรฐาน ใน C++ ถ้าไม่มี "using namespace std;" เราจะต้องพิมพ์ว่า *std::cout*. ซึ่งถูกนิยามไว้ใน *iostream* header file.
- "<<" เป็นตัวดำเนินการการแทรกข้อมูลใน C++ โดยใช้ส่งตัวอักษรจากด้านขวาไปยังวัตถุทางซ้าย ส่วน *endl* เป็นตัวอักษรขึ้นบรรทัดใหม่ใน C++

Header Files

Header file มีไว้เรียกใช้ Function หรือ Class ต่างๆ ที่เราต้องการใช้ในตอนเขียนโปรแกรม ส่วน `<iostream>` คือ Header file ที่ใช้สำหรับแสดงผลบนหน้าจอ (Console) หรือ ที่เรียกว่า **standard output** โดยสามารถใช้ function “`cout`” เพื่อแสดงผลข้อความ หรือค่าต่างๆ บนหน้าจอได้



```
#include <iostream>

using namespace std;

int main()
{
    string hello = "Hello";
    string world = "world!";
    string msg = hello + " "
+ world ;
    cout << msg << endl;
    msg[0] = 'h';
    cout << msg << endl;
    return 0;
}
```

Example Program

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

int main()
{
    double num1 = 5,
           num2, sum;
    num2 = 12;

    sum = num1 + num2;
    cout << "The sum is " << sum;
    return 0;
}
```

The Parts of a C++ Program

```
// sample C++ program      ← comment
#include <iostream>          ← preprocessor directive
using namespace std;        ← which namespace to use

int main()                  ← beginning of function named main
{                            ← beginning of block for main
    cout << "Hello, there!"; ← output statement
    return 0;               ← send 0 back to operating system
}                            ← end of block for main
```

The Parts of a C++ Program

Statement	Purpose
<code>// sample C++ program</code>	comment
<code>#include <iostream></code>	preprocessor directive
<code>using namespace std;</code>	namespace to use
<code>int main()</code>	beginning of function named <code>main</code>
<code>{</code>	beginning of block for <code>main</code>
<code> cout << "Hello, there!";</code>	output statement
<code> return 0;</code>	send 0 back to the operating system
<code>}</code>	end of block for <code>main</code>

Special Characters

Character	Name	Description
//	Double Slash	Begins a comment
#	Pound Sign	Begins preprocessor directive
< >	Open, Close Brackets	Encloses filename used in <code>#include</code> directive
()	Open, Close Parentheses	Used when naming function
{ }	Open, Close Braces	Encloses a group of statements
" "	Open, Close Quote Marks	Encloses string of characters
;	Semicolon	Ends a programming statement

Important Details

- C++ is case-sensitive. Uppercase และ lowercase นั่นคือ 'Main' ไม่เหมือนกับ 'main'.
- ทุกๆโปรแกรมต้องมี { ... }

The cout Object

- แสดงข้อมูลบนหน้าจอคอมพิวเตอร์
- ใช้ << เพื่อส่งข้อมูลไปยัง cout

```
cout << "Hello, there!";
```

- สามารถใช้ << เพื่อส่งหลายรายการไปยัง cout ได้

```
cout << "Hello, " << "there!";
```

หรือ

```
cout << "Hello, ";
```

```
cout << "there!";
```

Starting a New Line

- เพื่อให้ได้ผลลัพธ์หลายบรรทัดบนหน้าจอ
 - Use `endl`
`cout << "Hello, there!" << endl;`
 - Use `\n` in an output string
`cout << "Hello, there!\n";`

Escape Sequences – More Control Over Output

Escape Sequence	Name	Description
<code>\n</code>	Newline	Causes the cursor to go to the next line for subsequent printing.
<code>\t</code>	Horizontal tab	Causes the cursor to skip over to the next tab stop.
<code>\a</code>	Alarm	Causes the computer to beep.
<code>\b</code>	Backspace	Causes the cursor to back up, or move left one position.
<code>\r</code>	Return	Causes the cursor to go to the beginning of the current line, not the next line.
<code>\\</code>	Backslash	Causes a backslash to be printed.
<code>\'</code>	Single quote	Causes a single quotation mark to be printed.
<code>\"</code>	Double quote	Causes a double quotation mark to be printed.

#include Directive

- คำสั่งแรกในโปรแกรมเป็นการนำเข้า Standard Library ของภาษา C++
- ในตัวอย่าง เป็นการนำเข้าไลบรารี iostream ซึ่งจะประกอบไปด้วยฟังก์ชันการทำงานเกี่ยวกับ Input และ Output ซึ่งในภาษา C++ นั้นมีไลบรารีอื่นๆ อีกมากมาย สามารถนำเข้าไลบรารีเหล่านั้นได้ด้วย directive #include

- Example:

`#include <iostream>`

No ; goes here



Variables, Literals, and the Assignment Statement

- Variable

- ประกอบไปด้วยชื่อและประเภทของข้อมูล



- ใช้เพื่ออ้างอิงถึงตำแหน่งในหน่วยความจำที่สามารถเก็บค่าได้
- ต้องกำหนดก่อนที่จะใช้งานได้
- ค่าที่เก็บไว้สามารถเปลี่ยนแปลงได้

Variables

- หากมีการเก็บค่าใหม่ในตัวแปร ค่าก่อนหน้านั้นจะถูกแทนที่ด้วยค่าใหม่
- ค่าก่อนหน้านั้นถูกเขียนทับและไม่สามารถเรียกคืนได้อีกต่อไป

```
int age;  
age = 17;           // age is 17  
cout << age;        // Displays 17  
age = 18;           // Now age is 18  
cout << age;        // Displays 18
```


Constants

Literal

- ค่าที่ไม่เปลี่ยนแปลงระหว่างการ execute โปรแกรม
- เรียกว่า constant

'A' // character constant

"Hello" // string literal

12 // integer constant

3.14 // floating-point constant

Identifiers

- ชื่อที่กำหนดโดยโปรแกรมเมอร์ เพื่อใช้แทนส่วนต่างๆ ภายในโปรแกรม เช่น ตัวแปร `variables`
- ชื่อควรแสดงถึงการใช้งานของตัวแปร
- ไม่สามารถใช้ `Keyword` ของ `C++` เป็น `identifiers`
- `Identifiers` จะต้องขึ้นต้นด้วยตัวพิมพ์ใหญ่ หรือ พิมพ์เล็ก อาจตามด้วย `_` และตามด้วยตัวอักษร ตัวเลข หรือ `_`

Multi-word Variable Names

- ชื่อตัวแปรที่ใช้คำอธิบายอาจประกอบไปด้วยคำหลายคำ
- มีสองแนวทางในการตั้งชื่อตัวแปรดังนี้:
- ใช้ตัวพิมพ์ใหญ่ในอักษรแรกของคำ

quantityOnOrder

totalSales

- ใช้ underscore _ แทนช่องว่าง:

quantity_on_order

total_sales

Valid and Invalid Identifiers

IDENTIFIER	VALID?	REASON IF INVALID
<code>totalSales</code>	Yes	
<code>total_Sales</code>	Yes	
<code>total.Sales</code>	No	Cannot contain period
<code>4thQtrSales</code>	No	Cannot begin with digit
<code>totalSale\$</code>	No	Cannot contain \$

Integer Data Types

- Integer ถูกออกแบบเพื่อเก็บเลขจำนวนเต็ม ไม่มีทศนิยม
- สามารถเป็นได้ทั้งแบบ **signed** หรือ **unsigned**
12 -6 +3
- สามารถกำหนดให้มีขนาดที่แตกต่างกันได้ (*i.e.*, number of bytes): **short**, **int**, and **long**
- โดยขนาดของ **short** \leq size of **int** \leq size of **long**

Signed vs. Unsigned Integers

- C++ จะจัดสรร 1 bit สำหรับการกำหนดเครื่องหมายของตัวเลข และ bit ที่เหลือจะแทนข้อมูล
- ถ้าโปรแกรมที่เราสร้างไม่เคยที่จะเป็นลบเลย เราสามารถประกาศตัวแปรเป็น **unsigned** ได้เลย
- โดยปกติถ้าไม่มีการกำหนดอะไร ตัวแปรจะเป็น **signed**

Defining Variables

- ตัวแปรที่เป็น type เดียวกัน สามารถนิยามได้เป็น

- แบบคำสั่งแยก

```
int length;  
int width;
```

- อยู่ในคำสั่งเดียวกัน

```
int length,  
    width;
```

- ตัวแปรต่างชนิดกันจะต้องนิยามแยกกันเท่านั้น

Integer

- แบ่งออกเป็น 2 ประเภท คือ
 - `int` หรือ `short` เก็บเลขจำนวนเต็มตั้งแต่ -32,768 ถึง 32,767
 - `Long` เก็บเลขจำนวนเต็มตั้งแต่ -2,147,483,648 ถึง 2,147,483,647
 - วิธีการใช้คือถ้าต้องการตัวเลขจำนวนเต็มมากกว่า 32,767 เราจะต้องประกาศตัวแปรแบบ `long` ถ้าน้อยกว่าก็ประกาศแบบ `int` ดังตัวอย่าง
- `int a,b,c;`
 - `int age;`
 - `int height;`
 - `long salary,money;`

Floating-Point Data Types

- ออกแบบมาเพื่อทำการเก็บเลขจำนวนจริง

12.45

-3.8

- ทุกจำนวนเป็น signed
- มีขนาดให้เลือกหลายรูปแบบ
float, double, และ long double
- ขนาดของ **float** \leq size of **double**
 \leq size of **long double**

Floating-point Constants

- สามารถเขียนเป็น

- ทศนิยมปกติ:

31.4159

0.0000625

- รูปที่มี e :

3.14159E1

6.25e-5

- ค่าเริ่มต้นของตัวแปรที่เก็บจำนวนจริงใน C++ คือ double
- แต่สามารถกำหนดให้เป็น float ด้วยการใส่ตัวอักษร 'F' ต่อท้ายตัวเลข เช่น 3.14159F
- หรือกำหนดให้เป็น long double ด้วยการใส่ตัวอักษร 'L' ต่อท้ายตัวเลข เช่น 0.0000625L

Assigning Floating-point Values to Integer Variables

ถ้ามีการกำหนดค่าจำนวนจริงให้กับตัวแปรจำนวนเต็มใน C++ จะทำการตัดทศนิยมออก (หรือก็คือตัดเศษทิ้งไป) และใช้เฉพาะจำนวนเต็มเท่านั้น

ค่าจำนวนจริงจะไม่ถูกปัดเศษทิ้งไป แต่จะถูกตัดไปโดยทั้งหมด

```
int rainfall = 3.88;  
cout << rainfall;           // Displays 3
```

The char Data Type

- ใช้เพื่อเก็บตัวอักษรเดียว
- ใช้พื้นที่เก็บ 1 byte
- การเก็บในหน่วยความจำจะเก็บเป็นรหัสตัวเลขที่แทนตัวอักษร

SOURCE CODE

```
char letter = 'C';
```

MEMORY

```
letter
```



67

Character Literal

- เมื่ออ้างอิงถึงตัวอักษรสัญลักษณ์ดังกล่าวในโปรแกรม จะต้องครอบกลุ่มด้วยเครื่องหมายอัญประกาศเดี่ยว (single quotation marks) ดังนี้:
- `cout << 'Y' << endl;`
- เครื่องหมายอัญประกาศไม่ใช่ส่วนหนึ่งของตัวอักษรสัญลักษณ์เอง และจะไม่ถูกแสดงผลในการแสดงผลทางหน้าจอ

String Literals

- ข้อความ (string) สามารถเก็บได้เป็นชุดของตัวอักษรติดต่อกันในหน่วยความจำต่อเนื่องกันได้ เช่น "Hello"
- แต่ในการเก็บข้อมูลแบบนี้จะต้องมีการเพิ่มตัวอักษรหนึ่งตัวเพิ่มเติมเรียกว่า null terminator เพื่อบอกว่าตัวอักษรที่เก็บจบลงแล้ว ตัวอักษร null terminator จะมีค่าเป็น \0 และจะถูกเพิ่มอัตโนมัติต่อท้ายข้อความที่ถูกเก็บในหน่วยความจำ

H	e	l	l	o	\0
---	---	---	---	---	----

- ข้อความ (string) ประกอบด้วยตัวอักษรที่อยู่ระหว่างเครื่องหมายคำพูดคู่ (double quotation marks) " " เช่น "Hello World" เป็นต้น

The C++ `string` Class

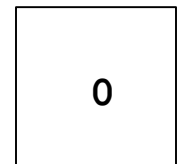
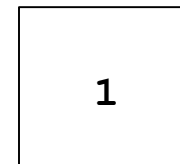
- `#include <string>` ใช้ในการสร้าง `string` object
- นิยามตัวแปร `string`
`string name;`
- สามารถกำหนดค่าของตัวแปรได้
`name = "George";`
- สามารถใช้ `cout` ในการแสดงผลลัพธ์ได้
`cout << "My name is " << name;`

The bool Data Type

- แทนค่าในรูปแบบ **true** หรือ **false**
- **bool** ค่าจะถูกเก็บในรูปแบบของ integer
- **false** ถูกแทนด้วย 0, **true** แทนด้วย 1

```
bool allDone = true;
bool finished = false;
```

allDone finished



Determining the Size of a Data Type

คำสั่ง **sizeof** เป็นตัวดำเนินการ (operator) ในภาษาโปรแกรม C++ ที่ใช้เพื่อหาขนาดของข้อมูลหรือตัวแปรใดๆ ในหน่วยความจำ (memory) โดยขนาดจะถูกตีความเป็นจำนวน **byte** ที่ใช้ในการเก็บข้อมูลนั้นๆ

ตัวอย่างการใช้ **sizeof operator** ในการหาขนาดของตัวแปรชนิด **int**:

```
int x;  
cout << "Size of int variable x is: " << sizeof(x) << " bytes\n";
```

ผลลัพธ์ที่ได้จะแสดงขนาดของตัวแปร **x** ในหน่วย **byte** บนหน้าจอเช่น
"Size of int variable x is: 4 bytes" (หมายความว่าตัวแปรชนิด **int** ใช้พื้นที่ในหน่วยความจำ 4 ไบต์)

Binary Arithmetic Operators

SYMBOL	OPERATION	EXAMPLE	ans
+	addition	ans = 7 + 3;	10
-	subtraction	ans = 7 - 3;	4
*	multiplication	ans = 7 * 3;	21
/	division	ans = 7 / 3;	2
%	modulus	ans = 7 % 3;	1

/ Operator

- C++ division operator (/) การหารเป็นจำนวนเต็ม ผลลัพธ์ของการหารจะเป็นจำนวนเต็มเสมอ

```
cout << 13 / 5;    // displays 2
cout <<  2 / 4;    // displays 0
```

- แต่ถ้าทั้งสองตัวดำเนินการเป็นจำนวนทศนิยม ผลการหารจะเป็นจำนวนทศนิยมเหมือนกัน

```
cout << 13 / 5.0;  // displays 2.6
cout << 2.0 / 4;   // displays 0.5
```

% Operator

- C++ modulus operator (%) คำนวณเอาเศษ

```
cout << 9 % 2;    // displays 1
```

- `cout << 9 % 2.0; // error`

- ต้องเป็น integer ทั้งคู่

Single-Line Comments

- Begin with `//` and continue to the end of line

```
int length = 12; // length in inches
int width = 15;  // width in inches
int area;        // calculated area

// Calculate rectangle area
area = length * width;
```

Multi-Line Comments

- Begin with `/*` and end with `*/`
- Can span multiple lines

```
/*-----  
    Here's a multi-line comment  
-----*/
```

- Can also be used as single-line comments

```
int area;    /* Calculated area */
```

Expressions and Interactivity

The `cin` Object

- มาตรฐาน input object
- ต้องการการเรียกใช้ `iostream` file เหมือนกับ `cout`
- ใช้อ่าน input จาก keyboard
- มักใช้กับ `cout` เพื่อแสดง prompt สำหรับการรับข้อมูล
- ข้อมูลดึงจาก `cin >>` เพื่อไปเก็บในตัวแปร

The `cin` Object

- `cin` สามารถใช้รับค่าหลาย ๆ ตัวแปรพร้อมกันได้
- `cin >> height >> width;`
- ค่าที่รับเข้ามาไม่จำเป็นต้องเป็นชนิดเดียวกันทั้งหมด แต่ลำดับของค่าจะมีความสำคัญ ค่าแรกที่ป้อนเข้ามาจะถูกเก็บไว้ในตัวแปรแรก ค่าถัดไปจะถูกเก็บไว้ในตัวแปรถัดไปตามลำดับ

Mathematical Expressions

- นิพจน์ (expression) สามารถเป็น
- ค่าคงที่ (constant),
- ตัวแปร (variable),
- หรือสมการทางคณิตศาสตร์ที่ประกอบด้วยค่าคงที่และตัวแปรซึ่งรวมกันด้วยตัวดำเนินการ (operator) ต่าง ๆ
- 2
 $height$
 $a + b / c$

Using Mathematical Expressions

- สามารถใช้ Mathematic Expression กับ คำสั่ง **cout** ได้

- Examples:

```
area = 2 * PI * radius;
```

```
cout << "border is: " << (2*(1+w)) ;
```

This is an
expression

These are
expressions

Order of Operations

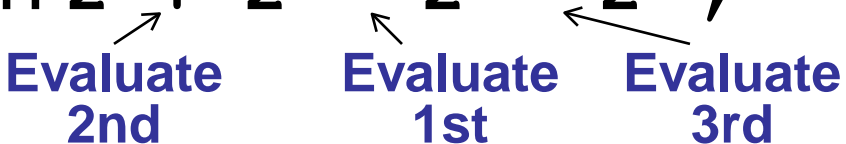
- ถ้า expression มี > 1 operator, จะต้องทำงานตามลำดับ

Do first: () expressions in parentheses

Do next: $-$ (unary negation) in order, left to right

Do next: $*$ $/$ $\%$ in order, left to right

Do last: $+$ $-$ in order, left to right

- In the expression $2 + 2 * 2 - 2$,

Evaluate 2nd Evaluate 1st Evaluate 3rd

Multiple and Combined Assignment

- การใช้ operator (=) สามารถใส่ได้หลายครั้งใน 1 expression

`x = y = z = 5;`

- Associates right to left

`x = (y = (z = 5)) ;`

The diagram illustrates the right-to-left evaluation of the combined assignment statement `x = y = z = 5;`. The expression is shown as `x = (y = (z = 5)) ;`. Three blue arrows point from the text labels below to the assignment operators in the code: the first arrow points from "Done 1st" to the operator between `z` and `5`; the second arrow points from "Done 2nd" to the operator between `y` and the opening parenthesis; and the third arrow points from "Done 3rd" to the operator between `x` and the opening parenthesis. This visualizes the sequence of operations from right to left.

More Examples

x += 5; means **x = x + 5;**

x -= 5; means **x = x - 5;**

x *= 5; means **x = x * 5;**

x /= 5; means **x = x / 5;**

x %= 5; means **x = x % 5;**

x *= a + b; means **x = x * (a + b);**

Working with Characters and Strings

- **char**: เก็บอักขระตัวเดียว
- **string**: เก็บสายของอักขระ

String Input

Reading in a string object

```
string str;
```

```
cin >> str;
```

```
// Reads in a string  
// with no blanks
```


String Operators

= การกำหนดค่าของ `string`

```
string words;  
words = "Tasty";
```

+ Joins two strings together

```
string s1 = "hot", s2 = "dog";  
string food = s1 + s2; // food = "hotdog"
```

+= Concatenates a string onto the end of another one

```
words += food; // words now = "Tasty hotdog"
```

String Member Functions

- **length()** – จำนวนอักขระใน string

```
string firstPrez = "George Washington";  
int size = firstPrez.length(); // size is  
17
```

- **assign()** – ใช้เมื่อต้องการกำหนดอักขระซ้ำๆ

```
string equals;  
equals.assign(80, '=');  
cout << equals << endl;
```

```
/* C++ Program to Calculate Multiplication of  
two Numbers */
```

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

```
int main()  
{  
    double first, second, product;  
  
    cout << "Enter 1st number :: ";  
    cin >> first;  
    cout << "\nEnter 2nd number :: ";  
    cin >> second;  
  
    product = first * second;  
  
    cout << "\nProduct of Two Numbers [  
"<<first<<" * "<<second<<" ] = " <<  
product<<"\n";  
  
    return 0;  
}
```

Enter 1st number :: 23

Enter 2nd number :: 12

Product of Two Numbers [23 * 12] = 276

```
/* C++ Program to Find Sum and Average of three numbers */
```

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

```
int main()
{
    float a,b,c,sum,avg;
    cout<<"Enter 1st number :: ";
    cin>>a;
    cout<<"\nEnter 2nd number :: ";
    cin>>b;
    cout<<"\nEnter 3rd number :: ";
    cin>>c;
```

```
    sum=a+b+c;
```

```
    avg=sum/3;
```

```
    cout<<"\nThe SUM of 3 Numbers [ "<<a<<" + "<<b<<" + "<<c<<" ] =
"<<sum<<"\n";
```

```
    cout<<"\nThe AVERAGE of 3 Numbers [ "<<a<<,"<<b<<,"<<c<<" ] =
"<<avg<<"\n";
```

```
    return 0;
```

```
}
```

Enter 1st number :: 12

Enter 2nd number :: 22

Enter 3rd number :: 33

The SUM of 3 Numbers [12 + 22 + 33] = 67

The AVERAGE of 3 Numbers [12, 22, 33] = 22.3333

```
#include<iostream>
using namespace std;
const int NUMBER = 12;
int main()
{
    int firstNum;
    int secondNum;
    firstNum = 18;
    cout<<"Firstnum = "<<firstNum<<endl;
    cout<<"Enter an integer : ";
    cin >> secondNum;
    cout<<"Secondnum = " <<secondNum << endl;
    firstNum = firstNum+NUMBER+2*secondNum;
    cout<<"The new value of ";
    firstNum = firstNum+NUMBER+2*secondNum;
    cout<< "The new value of "
        <<"Firstnum = "<< firstNum << endl;
    return 0;
}
```

```
Firstnum = 18
Enter an integer : 2
Secondnum = 2
The new value of The new value of Firstnum = 50
```

```
#include<iostream>
using namespace std;
const double CENTEMETERS_PER_INCH = 2.54;
const int INCHES_PER_FOOT = 12;
int main()
{Enter two integers, one for feet and one for inches: 6 8
```

The numbers you entered are 6 for feet and 8 for inches.

The total number of inches = 80

The number of centimeters = 203.2

```
int feet, inches;
int totalInches;
double centimeter;
```

Enter two integers, one for feet and one for inches: 6 8

The numbers you entered are 6 for feet and 8 for inches.

The total number of inches = 80

The number of centimeters = 203.2

```
cout << "Enter two integers, one for feet and " << "one for inches: ";
```

```
cin >> feet >> inches;
```

```
cout << endl;
```

```
cout << "The numbers you entered are " << feet
```

```
    << " for feet and " << inches
```

```
    << " for inches. " << endl;
```

```
totalInches = INCHES_PER_FOOT*feet+inches;
```

```
cout << "The total number of inches = "
```

```
    << totalInches << endl;
```

```
centimeter = CENTEMETERS_PER_INCH * totalInches;
```

```
cout << "The number of centimeters = "
```

```
    << centimeter << endl;
```

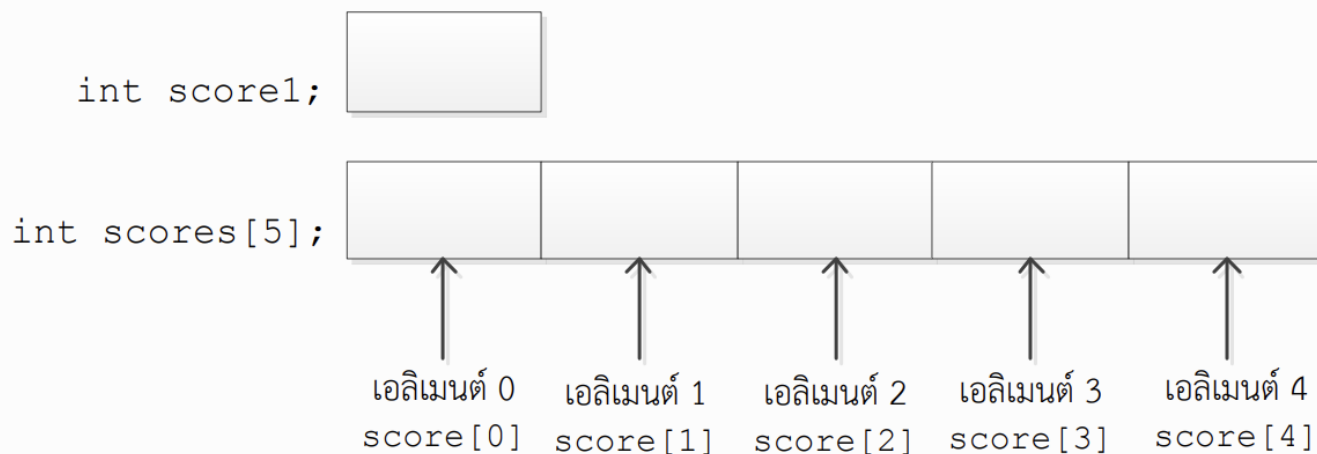
```
return 0;
```

```
}
```

ARRAY

Arrays Hold Multiple Values

- Array ใช้เก็บข้อมูลเป็นกลุ่ม โดยข้อมูลจะต้องมี data type เดียวกัน
- เช่น ต้องการเก็บคะแนนของ น.ศ. จำนวน 5 คน อาจเลือกใช้
 - score1, score2, score3, score4, score5
 - scores[5] -> จอง พ.ท. หน่วยความจำติดกันไว้ให้ 5 ห้อง



Array Variable

Data type Name of array [size of array]

- Data type – กำหนด data type ได้ชนิดเดียวเท่านั้น
- Name of array – ตามหลักการกำหนดชื่อของตัวแปร
- Size of array – จ.น. ข้อมูล หรือ เอลิเมนต์(element) ที่ต้องการจองพ.ท.

• การประกาศใช้งาน array `int sale[] = {100,294,244,500};`

- บอก จ.น. ข้อมูล
- ไม่บอก จ.น. ข้อมูล

การประกาศอาเรย์	จำนวนเอลิเมนต์	หน่วยความจำสำหรับชนิดข้อมูล 1 ตัว (ไบต์)	หน่วยความ ทั้งหมด (ไบต์)
<code>int scores[5];</code>	5	4	20
<code>float average[100];</code>	100	4	400
<code>double income[12];</code>	12	8	96
<code>char name[10];</code>	10	1	10

Array Terminology Examples

Examples:

สมมติให้ **int** ใช้ 4 bytes และ **double** ใช้ 8 bytes

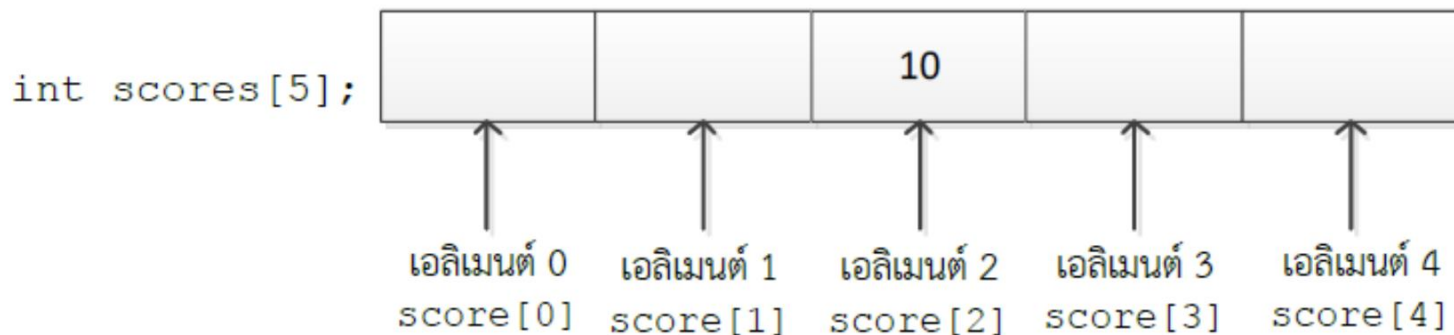
```
const int ISIZE = 5, DSIZE = 10;
```

```
int tests[ISIZE]; // holds 5 ints, array  
                  // occupies 20 bytes
```

```
double volumes[DSIZE]; // holds 10 doubles,  
                        // array occupies  
                        // 80 bytes
```

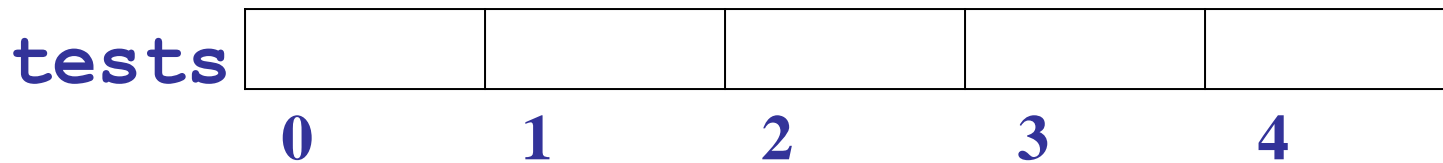
Accessing Array Elements

- เข้าถึงในแต่ละเอलिเมนต์
- การอ้างอิงจะใช้ `index` ที่จะเริ่มเอलिเมนต์แรกคือ 0 เสมอ และเอลิเมนต์สุดท้าย คือ $n-1$; n คือ ขนาดของอาร์เรย์
- การกำหนดค่าให้กับเอลิเมนต์ในอาร์เรย์
- – `scores[2] = 10`



Accessing Array Elements

สมาชิกของ Array (จะถูกเข้าถึงโดย array name และ subscript)



```
tests[0] = 79;
```

```
cout << tests[0];
```

```
cin >> tests[1];
```

```
tests[4] = tests[0] + tests[1];
```

Inputting and Displaying Array Contents

cout และ **cin** สามารถนำมาใช้ในการ
แสดงผลและเก็บค่าของ Array ได้

```
const int ISIZE = 5;  
  
int tests[ISIZE]; // Define 5-elt. array  
cout << "Enter first test score ";  
cin  >> tests[0];
```

Array Subscripts

- Array subscript สามารถเป็น integer constant, integer variable, หรือ integer expression

- Examples:

Subscript is

`cin >> tests[3];` int constant

`cout << tests[i];` int variable

`cout << tests[i+j];` int expression

Accessing All Array Elements

การเข้าถึงสมาชิกใน array

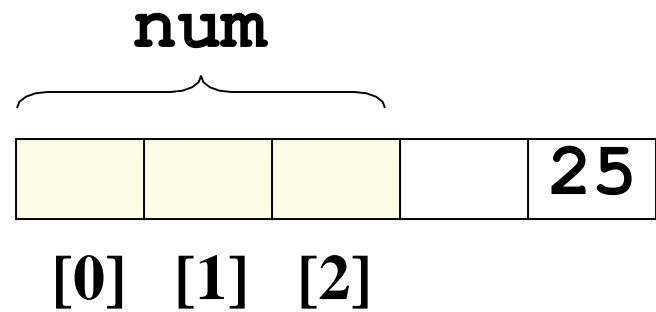
- ใช้ loop
- ให้ loop ควบคุมตัวแปรที่เป็น array subscript

```
for (i = 0; i < 5; i++)  
    cout << tests[i] << endl;
```

No Bounds Checking

- ใน C++ จะไม่มีการตรวจเช็ค array subscript ที่ถูกต้อง
- ถ้ามีการใช้ array subscript ที่ผิด สามารถเขียนโปรแกรมทับลงในหน่วยความจำได้เลย
- Example:

```
const int ISIZE = 3;  
int i = 4;  
int num[ISIZE];  
num[i] = 25;
```



Array Initialization

- สามารถเริ่มต้นในระหว่างการดำเนินการโปรแกรมได้

```
tests[0] = 79;  
tests[1] = 82; // etc.
```

- สามารถเริ่มต้นได้ที่กำหนดอาร์เรย์ตั้งแต่เริ่มต้น
- ```
const int ISIZE = 5;
int tests[ISIZE] = {79, 82, 91, 77, 84};
```

# Partial Array Initialization

- การกำหนดค่า array เริ่มต้น ถ้ามีการกำหนดค่าน้อยกว่าขนาดที่ประกาศไว้ใน array ค่าสมาชิกที่เหลือจะถูกตั้งค่าให้เป็น 0 หรือ เป็น empty string

```
int tests[ISIZE] = {79, 82};
```

|    |    |   |   |   |
|----|----|---|---|---|
| 79 | 82 | 0 | 0 | 0 |
|----|----|---|---|---|

# Implicit Array Sizing

- สามารถกำหนดขนาดของ array ได้โดยใช้ขนาดของรายการข้อมูลเริ่มต้น

```
short quizzes[]={12,17,15,11};
```

|    |    |    |    |
|----|----|----|----|
| 12 | 17 | 15 | 11 |
|----|----|----|----|

# Using Increment and Decrement Operators with Array Elements

When using ++ and -- operators, don't confuse the element with the subscript

```
tests[i]++; // adds 1 to tests[i]
tests[i++]; // increments i, but has
 // no effect on tests
```

# Copying One Array to Another

- ไม่สามารถกำหนดคำสั่งแบบนี้เพื่อ copy array ได้

```
tests2 = tests; //won't work
```

- ต้องใช้ loop ในการ copy ค่าของสมาชิก แบบ element- by-element:

```
for (int indx=0; indx < ISIZE; indx++)
 tests2[indx] = tests[indx];
```

# Sum, Average of Array Elements

- ใช้ loop เพื่อ add ค่าภายใน array element ไปด้วยกัน

```
float average, sum = 0;
for (int tnum=0; tnum< ISIZE; tnum++)
 sum += tests[tnum];
```

- หลังจากได้ค่า sum, คำนวณค่า average  

```
average = sum/ISIZE;
```

# Largest Array Element

- ใช้ loop ในการตรวจสอบเพื่อหา ค่า largest element (*i.e.*, one with the largest value)

```
int largest = tests[0];
for (int tnum = 1; tnum < ISIZE; tnum++)
{ if (tests[tnum] > largest)
 largest = tests[tnum];
}
cout << "Highest score is " << largest;
```

- smallest element ทำเหมือนกัน

# C-Strings and `string` Objects

```
string city;
cout << "Enter city name: ";
cin >> city;
```

|     |     |     |     |     |
|-----|-----|-----|-----|-----|
| 's' | 'a' | 'l' | 'e' | 'm' |
|-----|-----|-----|-----|-----|

city[0] city[1] city[2] city[3] city[4]

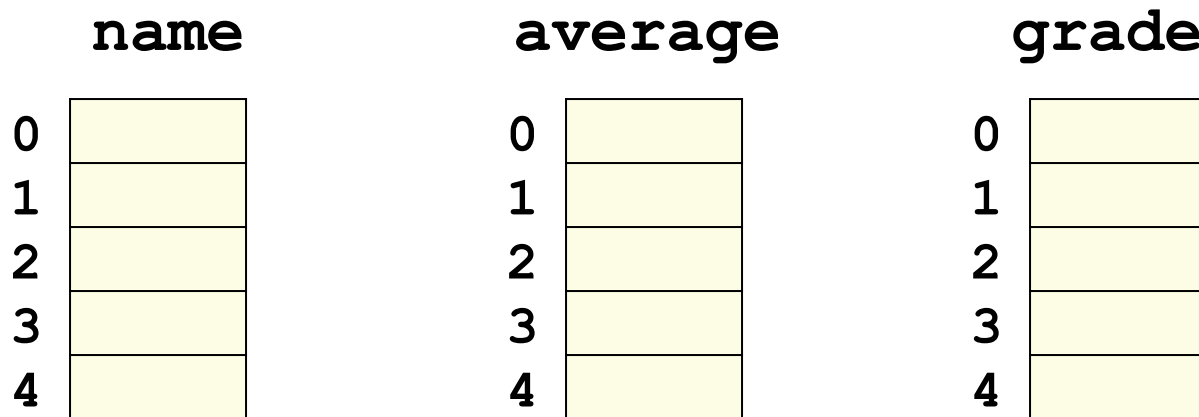


# Using Parallel Arrays

- **Parallel arrays:** array ตั้งแต่สองอันขึ้นไป ที่เก็บค่าที่สัมพันธ์กัน
- Subscript เพื่อเชื่อมโยง arrays
  - elements ที่มี subscript เดียวกันจะสัมพันธ์กัน
- โดยอาร์เรย์แต่ละอันไม่จำเป็นต้องเก็บข้อมูลประเภทเดียวกัน

# Parallel Array Example

```
const int ISIZE = 5;
string name[ISIZE]; // student name
float average[ISIZE]; // course average
char grade[ISIZE]; // course grade
```



# Parallel Array Processing

```
const int ISIZE = 5;
string name[ISIZE]; // student name
float average[ISIZE]; // course average
char grade[ISIZE]; // course grade
...
for (int i = 0; i < ISIZE; i++)
 cout << " Student: " << name[i]
 << " Average: " << average[i]
 << " Grade: " << grade[i]
 << endl;
```

# 2D Array Traversal

- To initialize
  - Default of 0
  - Initializers grouped by row in braces

```
int b[2][2] = { { 1, 2 }, { 3, 4 } };
```

Row 0                  Row 1

|   |   |
|---|---|
| 1 | 2 |
| 3 | 4 |

```
int b[2][2] = { { 1 }, { 3, 4 } };
```

|   |   |
|---|---|
| 1 | 0 |
| 3 | 4 |

```
// Fig. 4.22: fig04_22.cpp
// Initializing multidimensional arrays.
```

```
#include <iostream>
```

```
using std::cout;
```

```
using std::endl;
```

```
void printArray(int [][] 3);
```

```
int main()
```

```
{
```

```
 int array1[2][3] = { { 1, 2, 3 }, { 4, 5, 6 } };
```

```
 int array2[2][3] = { 1, 2, 3, 4, 5 };
```

```
 int array3[2][3] = { { 1, 2 }, { 4 } };
```

```
 cout << "Values in array1 by row are:" << endl;
```

```
 printArray(array1);
```

```
 cout << "Values in array2 by row are:" << endl;
```

```
 printArray(array2);
```

```
 cout << "Values in array3 by row are:" << endl;
```

```
 printArray(array3);
```

```
 return 0; // indicates successful termination
```

```
} // end main
```

Note the format of the prototype.

Note the various initialization styles. The elements in **array2** are assigned to the first row and then the second.

```
// function to output array with two rows
void printArray(int a[][3])
{
 for (int i = 0; i < 2; i++) { // f

 for (int j = 0; j < 3; j++) // output column values
 cout << a[i][j] << ' ';

 cout << endl; // start new line of output

 } // end outer for structure
} // end function printArray
```

For loops are often used to iterate through arrays. Nested loops are helpful with multiple-subscripted arrays.

# Program to print one dimensional array

```
#include<iostream>
using namespace std;
int main()
{
 int arr[50], num, i;
 cout<<"\n How Many Elements You Want to Store into an Array? \n";
 cin>>num;
 cout<<"\n Enter "<<num<<" Elements to Store into an Array : \n";
 for(i=0; i<num; i++)
 {
 cin>>arr[i];
 }
 cout<<"\n The Elements in the Array are : \n";
 for(i=0; i<num; i++)
 {
 cout<<arr[i]<<"\t";
 }
 return 0;
}
```

How Many Elements You Want to Store into an Array?  
4

Enter 4 Elements to Store into an Array  
:  
3  
4  
5  
6

The Elements in the Array are :  
3   4   5   6

# Program to calculate arithmetic mean of numbers

```
#include<iostream>
using namespace std;
int main()
{
 int num, i, arr[50], sum=0;
 cout<<"\n How Many Numbers You Want to Enter? \n";
 cin>>num;
 cout<<"\n Enter "<<num<<" Numbers : \n";
 for(i=0; i<num; i++)
 {
 cin>>arr[i];
 sum=sum+arr[i];
 }
 int armean=sum/num;
 cout<<"\n Arithmetic Mean = "<<armean;
 return 0;
}
```

How Many Numbers You Want to Enter?

4

Enter 4 Numbers :

5

6

7

8

Arithmetic Mean = 6



# Calculate average and percentage marks of a student

```
#include<iostream>
using namespace std;
int main()
{
 int marks[5], i;
 float sum=0;
 cout<<"\n Enter Marks of Student \n";
 cout<<"\n Thai : ";
 cin>>marks[0];
 cout<<"\n English : ";
 cin>>marks[1];
 cout<<"\n Maths : ";
 cin>>marks[2];
 cout<<"\n History : ";
 cin>>marks[3];
 cout<<"\n Science : ";
 cin>>marks[4];
```

```
for(i=0;i<5;i++)
{
 sum=sum+masks[i];
}

float avg=sum/5;
float per;
per=(sum/500)*100;
cout<<"\n Average Marks = "<<avg;
cout<<"\n Percentage = "<<per<<" %";
return 0;
}
```

Enter Marks of Student

Thai : 89

English : 76

Maths : 56

History : 98

Science : 68

Average Marks = 77.4

Percentage = 77.4 %

# Calculate grade of a student on the basis of his/her total marks

```
#include<iostream>
using namespace std;
int main()
{
 int marks[5], i;
 float sum=0,avg;

 cout<<"\n Enter Marks of Student \n";
 cout<<"-----";
 cout<<"\n Thai : ";
 cin>>marks[0];
 cout<<"\n English : ";
 cin>>marks[1];
 cout<<"\n Maths : ";
 cin>>marks[2];
 cout<<"\n History : ";
 cin>>marks[3];
 cout<<"\n Science : ";
 cin>>marks[4];
```

Enter Marks of Student

---  
Thai : 67

English : 76

Maths : 89

History : 78

Science : 56

---  
Total Marks of Student  
= 366

Average = 73.2

Grade = B

```
for(i=0;i<5;i++)
 { sum=sum+marks[i]; }
cout<<"-----";
cout<<"\n Total Marks of Student =
"<<sum;
 avg=sum/5;
 cout<<"\n Average = "<<avg;
 cout<<"\n Grade = ";

 if(avg>80)
 { cout<<"A"; }
 else if(avg>60 && avg<=80)
 { cout<<"B"; }
 else if(avg>40 && avg<=60)
 { cout<<"C"; }
 else
 { cout<<"D"; }
 return 0;
}
```

# Search an element in array

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

int main()
{
 int arr[10], i, num, n, cnt=0, pos;
 cout<<"\n Enter Array Size : ";
 cin>>n;
 cout<<"\n Enter Array Elements : \n";
 for(i=0; i<n; i++)
 {
 cout<<" ";
 cin>>arr[i];
 }
 cout<<"\n Enter Element to be Searched : ";
 cin>>num;
```

Enter Array Size : 3

Enter Array Elements :

23

44

55

Enter Element to be Searched : 23

Element 23 Found At Position 1

```
for(i=0; i<n; i++)
{
 if(arr[i]==num)
 {
 cnt=1;
 pos=i+1;
 break;
 }
}
if(cnt==0)
{
 cout<<"\n Element Not Found..!!";
}
else
{
 cout<<"\n Element "<<num<<" Found
At Position "<<pos;
}
return 0;
}
```

```
/* C++ Program to Find Largest and Smallest Element
of a Matrix */
```

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int m,n,a[10][10],i,j,high,low;
```

```
 cout<<"Enter no. of rows :: ";
```

```
 cin>>m;
```

```
 cout<<"\nEnter no. of coloumns :: ";
```

```
 cin>>n;
```

```
 cout<<"\nEnter Elements to Matrix Below :: \n";
```

```
 for(i=0;i<m;i++)
```

```
 {
```

```
 for(j=0;j<n;++j)
```

```
 {
```

```
 cout<<"\nEnter a["<<i<<"]["<<j<<"] Element :: ";
```

```
 cin>>a[i][j];
```

```
 }
```

```
 }
```

```
 cout<<"\nThe given matrix is :: \n\n";
```

```
 for (i = 0; i < m; ++i)
```

```
 {
```

```
 for (j = 0; j < n; ++j)
```

```
 {
```

```
 cout<<"\t"<<a[i][j];
```

```
 }
```

```
 printf("\n\n");
```

```
 }
```

```
 high=a[0][0];
```

```
 low=a[0][0];
```

```
 for(i=0;i<m;++i)
```

```
 {
```

```
 for(j=0;j<n;++j)
```

```
 {
```

```
 if(a[i][j]>high)
```

```
 high=a[i][j];
```

```
 else
```

```
 if(a[i][j]<low)
```

```
 low=a[i][j];
```

```
 }
```

```
 }
```

```
 cout<<"\nHighest Element ::
```

```
"<<high<<"\n\nLowest Element ::
```

```
"<<low<<"\n";
```

```
 return 0;
```

```
}
```

Pointer

# Pointers and the Address Operator

- แต่ละตัวแปรในโปรแกรมจะถูกเก็บไว้ในหน่วยความจำที่ถูกอ้างอิงตำแหน่งโดย Address
- ใช้ operator & เพื่อให้ได้ Address ของตัวแปร
- ```
int num = 23;  
cout << &num; // prints address  
               // in hexadecimal
```
- โดย address ของ memory location ถูกชี้โดยตัวแปร **pointer**

Pointer Variables

- Definition:

```
int    *intptr;           //ตัวแปร pointer
```

intptr จะเก็บ address ของตัวแปรประเภท integer ได้
หรือ ตัวแปร **intptr** ชี้ไปที่ตัวแปรประเภท **int** นั้นเอง

- ช่องว่างในการเว้นวรรคไม่สำคัญ:

```
int * intptr;  
int*  intptr;
```

- * เรียกว่าเครื่องหมาย **indirection operator**

Pointer Variables

- Assignment:

```
int num = 25;  
int *intptr;  
intptr = &num;
```



- Memory layout:

address of num: 0x4a00

- สามารถเข้าถึง **num** ได้โดยการใช้ **intptr** และ indirection operator *****:

```
cout << intptr;           // prints 0x4a00  
cout << *intptr;          // prints 25  
*intptr = 20;             // puts 20 in num
```


The Relationship Between Arrays and Pointers

ใช้ชื่อของ array ในการหา address เริ่มต้นของ array

```
int vals[] = {4, 7, 11};
```

4	7	11
---	---	----

starting address of vals: 0x4a00

```
cout << vals;    // displays 0x4a00  
cout << vals[0]; // displays 4
```

The Relationship Between Arrays and Pointers

- ดังนั้นชื่อของ array สามารถถูกใช้เป็น pointer ได้
- ```
int vals[] = {4, 7, 11};
cout << *vals; // displays 4
```
- pointer สามารถถูกใช้เป็นชื่อ array ได้  

```
int *valptr = vals;
cout << valptr[1]; // displays 7
```

# Pointers in Expressions

- ถ้าให้ :

```
int vals[]={4,7,11};
int *valptr = vals;
```

- อะไรคือ **valptr + 1**?
- นั่นคือ (address in **valptr**) + (1 \* size of an **int**)

```
cout << *(valptr+1); // displays 7
cout << *(valptr+2); // displays 11
```

# Array Access

Array elements can be accessed in many ways

| Array access method                       | Example                        |
|-------------------------------------------|--------------------------------|
| array name and [ ]                        | <code>vals[2] = 17;</code>     |
| pointer to array and [ ]                  | <code>valptr[2] = 17;</code>   |
| array name and subscript arithmetic       | <code>*(vals+2) = 17;</code>   |
| pointer to array and subscript arithmetic | <code>*(valptr+2) = 17;</code> |

# Array Access

- Array notation

`vals[i]`

is equivalent to the pointer notation

`*(vals + i)`

# Pointer Arithmetic

Some arithmetic operators can be used with pointers:

- Increment and decrement operators `++`, `--`
- Integers can be added to or subtracted from pointers using the operators `+`, `-`, `+=`, and `-=`
- One pointer can be subtracted from another by using the subtraction operator `-`

# Pointer Arithmetic

Assume the variable definitions

```
int vals[]={4,7,11};
```

```
int *valptr = vals;
```

Examples of use of ++ and --

```
valptr++; // points at 7
```

```
valptr--; // now points at 4
```

# More on Pointer Arithmetic

Assume the variable definitions:

```
int vals[]={4,7,11};
```

```
int *valptr = vals;
```

Example of the use of + to add an int to a pointer:

```
cout << *(valptr + 2)
```

This statement will print 11



# More on Pointer Arithmetic

Assume the variable definitions:

```
int vals[]={4,7,11};
```

```
int *valptr = vals;
```

Example of use of +=:

```
valptr = vals; // points at 4
```

```
valptr += 2; // points at 11
```

# Initializing Pointers

- Can initialize to NULL or 0 (zero)

```
int *ptr = NULL;
```

- Can initialize to addresses of other variables

```
int num, *numPtr = #
```

```
int val[ISIZE], *valptr = val;
```

- Initial value must have correct type

```
float cost;
```

```
int *ptr = &cost; // won't work
```

# Comparing Pointers

```
if (ptr1 == ptr2) // compares
 // addresses
if (*ptr1 == *ptr2) // compares
 // contents
```

```
#include<iostream>
```

```
using namespace std;
```

```
main()
```

```
{
```

```
 int number[] ={10,20,30,40,50};
```

```
 int *ptr = number;
```

```
 cout << ptr[1]<<endl;
```

```
 return 0;
```

```
}
```

20

```
#include<iostream>
```

```
using namespace std;
```

```
main()
```

```
{
 int numbers[20] = {10,20,30,40,50};
 int *ptr1 = numbers;
 cout << *ptr1<<endl;
 ptr1++;
 cout<< *ptr1<<endl;
 ptr1--;
 cout<< *ptr1;
 return 0;
}
```



```
10
20
10
```

```
#include<iostream>
```

```
using namespace std;
```

```
main()
```

```
{
```

```
 int numbers[] = {10,20,30,40,50};
```

```
 cout<<"The first element in the array is : ";
```

```
 cout<<*numbers<<endl;
```

```
 cout<<"The forth element in the array is : ";
```

```
 cout<< numbers[3]<<endl;
```

```
 cout<<"The forth element in the array is : ";
```

```
 cout<<*(numbers+3);
```

```
 return 0;
```

The first element in the array is : 10

The forth element in the array is : 40

The forth element in the array is : 40

```
#include<iostream>
```

```
using namespace std;
```

```
int main(){
```

```
 int x = 10;
```

```
 int* px ;
```

```
 px = &x;
```

```
 cout << "Value of x var : " << x << endl;
```

```
 cout << "Value of x (&x) : " << &x << endl;
```

```
 cout << "Value of px var : " << px << endl;
```

```
 cout << "Value of px (&px) : " << &px << endl;
```

```
 cout << "Value of *px : " << *px << endl;
```

```
 return 0;
```

```
}
```

Value of x var : 10

Value of x (&x) : 0x61fe1c

Value of px var : 0x61fe1c

Value of px (&px) : 0x61fe10

Value of \*px : 10

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

```
int main () {
 // an array with 5 elements.
 double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
 double *p;

 p = balance;

 // output each array element's value
 cout << "Array values using pointer " << endl;

 for (int i = 0; i < 5; i++) {
 cout << "*(p + " << i << ") : ";
 cout << *(p + i) << endl;
 }
 cout << "Array values using balance as address " << endl;

 for (int i = 0; i < 5; i++) {
 cout << "*(balance + " << i << ") : ";
 cout << *(balance + i) << endl;
 }
 return 0;
}
```

Array values using pointer

\*(p + 0) : 1000

\*(p + 1) : 2

\*(p + 2) : 3.4

\*(p + 3) : 17

\*(p + 4) : 50

Array values using balance as address

\*(balance + 0) : 1000

\*(balance + 1) : 2

\*(balance + 2) : 3.4

\*(balance + 3) : 17

\*(balance + 4) : 50



```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 float arr[3] {5,6,7};
```

```
 float *ptr;
```

```
 cout << "Display address using array : " << endl;
```

```
 for (int i=0; i<3; i++)
```

```
 {
```

```
 cout << i << " Array " << arr[i] << " Pointer " <<&arr[i] << endl;
```

```
 }
```

```
 return 0;
```

```
}
```

Display address using array :

0 Array 5 Pointer 0x61fe10

1 Array 6 Pointer 0x61fe14

2 Array 7 Pointer 0x61fe18

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
 int arr[5] {8,7,6,5,10};
```

```
 cout << *arr;
```

```
 char fname[] {"Weenawadee"};
```

```
 string name[] {"Weenawadee"};
```

```
 cout << *name<<"\n";
```

```
 cout << *fname;
```

```
 return 0;
```

```
}
```

8Weenawadee

W

# Vectors

- Holds a set of elements, like an array
- Flexible number of elements - can grow and shrink
  - No need to specify size when defined
  - Automatically adds more space as needed
- Defined in the Standard Template Library (STL)
  - Covered in a later chapter
- Must include **vector** header file to use vectors

```
#include <vector>
```

# Vectors

- Can hold values of any type
    - Type is specified when a vector is defined
- ```
vector<int> scores;  
vector<double> volumes;
```
- Can use `[]` to access elements

Defining Vectors

- Define a vector of integers (starts with 0 elements)
`vector<int> scores;`
- Define `int` vector with initial size 30 elements
`vector<int> scores(30);`
- Define 20-element `int` vector and initialize all elements to 0
`vector<int> scores(20, 0);`
- Define `int` vector initialized to size and contents of vector `finals`
`vector<int> scores(finals);`

Growing a Vector's Size

- Use **push_back** member function to add an element to a full array or to an array that had no defined size

```
// Add a new element holding a 75  
scores.push_back(75);
```

- Use **size** member function to determine number of elements currently in a vector

```
howbig = scores.size();
```

Removing Vector Elements

- Use **pop_back** member function to remove last element from vector
`scores.pop_back();`
- To remove all contents of vector, use **clear** member function
`scores.clear();`
- To determine if vector is empty, use **empty** member function
`while (!scores.empty()) ...`

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    // initializer list
    vector<int> vector1 = {1, 2, 3, 4, 5};
    // uniform initialization
    vector<int> vector2{6, 7, 8, 9, 10};
    // method 3
    vector<int> vector3(5, 12);
    cout << "vector1 = ";
    // ranged loop
    for (const int& i : vector1) {
        cout << i << " ";
    }
    cout << "\nvector2 = ";
    // ranged loop
    for (const int& i : vector2) {
        cout << i << " ";
    }
    cout << "\nvector3 = ";
    // ranged loop
    for (int i : vector3) {
        cout << i << " ";
    }
    return 0;
}
```

```
vector1 = 1 2 3 4 5
vector2 = 6 7 8 9 10
vector3 = 12 12 12 12 12
```


Add Elements to a Vector

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> num {1, 2, 3, 4, 5};
    cout << "Initial Vector: ";
    for (const int& i : num) {
        cout << i << " ";
    }
    // add the integers 6 and 7 to the vector
    num.push_back(6);
    num.push_back(7);
    cout << "\nUpdated Vector: ";
    for (const int& i : num) {
        cout << i << " ";
    }
    return 0;
}
```

Initial Vector: 1 2 3 4 5

Updated Vector: 1 2 3 4 5 6 7

Access Elements of a Vector

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

int main() {
    vector<int> num {1, 2, 3, 4, 5};

    cout << "Element at Index 0: " << num.at(0) << endl;
    cout << "Element at Index 2: " << num.at(2) << endl;
    cout << "Element at Index 4: " << num.at(4);

    return 0;
}
```

```
Element at Index 0: 1
Element at Index 2: 3
Element at Index 4: 5
```

Change Vector Element

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> num {1, 2, 3, 4, 5};
    cout << "Initial Vector: ";
    for (const int& i : num) {
        cout << i << " ";
    }
    // change elements at indexes 1 and 4
    num.at(1) = 9;
    num.at(4) = 7;
    cout << "\nUpdated Vector: ";
    for (const int& i : num) {
        cout << i << " ";
    }
    return 0;
}
```

Initial Vector: 1 2 3 4 5
Updated Vector: 1 9 3 4 7

Delete Elements from C++ Vectors

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> prime_numbers{2, 3, 5, 7};

    // initial vector
    cout << "Initial Vector: ";
    for (int i : prime_numbers) {
        cout << i << " ";
    }
    // remove the last element
    prime_numbers.pop_back();
    // final vector
    cout << "\nUpdated Vector: ";
    for (int i : prime_numbers) {
        cout << i << " ";
    }
    return 0;
}
```

Initial Vector: 2 3 5 7
Updated Vector: 2 3 5

C++ Vector Iterators

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<int> num {1, 2, 3, 4, 5};
    // declare iterator
    vector<int>::iterator iter;
    // initialize the iterator with the first
    element
    iter = num.begin();
    // print the vector element
    cout << "num[0] = " << *iter << endl;
    // iterator points to the 3rd element
    iter = num.begin() + 2;
    cout << "num[2] = " << *iter;
    // iterator points to the last element
    iter = num.end() - 1;
    cout << "num[4] = " << *iter;
    return 0;
}
```

```
num[0] = 1
num[2] = 3
num[4] = 5
```