Introduction to C++

Kornchawal Chaipah, PhD

Computer Engineering Department,
Khon Kaen University

Agenda

- C++ structure
- Output operators
- Variables and data types
- Arithmetic operators and precedence
- Type casting
- Input operators

C++ File and Compiler

- C++ file extension is .cpp
  - To let a compiler correctly interpret codes
- Compiler
  - A program to interpret a high level-language, e.g. C++
  - Human-readable codes -> machine readable codes
- Codeblock: an official programming environment for this class
- GNU C++: an official compiler for this class (already comes with Codeblock)

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

- Preprocessing directive
- A compiler will know where to get an object's definition (how to process/interpret an object)
- Standard C++ libraries and User-defined libraries

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

- Ask a program to look in the namespace std if it can't find a definition for any objects
- Write cout instead of std::cout
- Write endl instead of std::endl

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

- Use // to make notes to explain your code (single line comment)
- A compiler will ignore everything after // on that line

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
/* P1
   Kornchawal Chaipah
   ID: 583040111-1
*/
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World\n";
    return 0;
}
```

- Use /* .. */ to make notes to explain your code (multiline comment)
- A compiler will ignore everything within the /* .. */ pair

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

A statement is an order telling a computer what to do

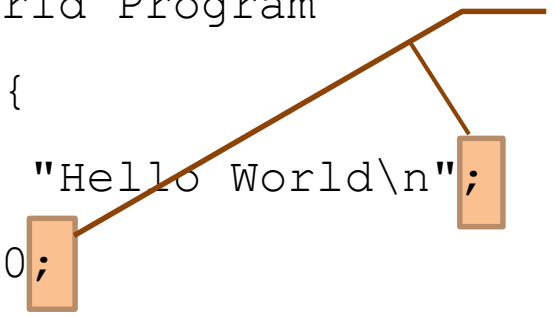
C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```



- Every statement must be ended with a semicolon (;)
- To tell a compiler that a statement is finished

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

All statements must be inside a pair of curly brackets {….} of the main() function

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

- C++ is case-sensitive
- E.g. Cout ≠ cout, X ≠ x

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    return 0;
}
```

- main() will return an integer to an OS when finish running
- Usually return 0 to indicate that a program successfully ends (normal termination)
- If return non-zero, usually something is wrong

C++ Basic Structure

- Every program must have a main() function

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

- main() is a starting point when running a program

```
#include <iostream>
#include <cstdlib>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    exit(1);
}
```

- exit() is also used to terminate an application
- must include <cstdlib>

C++ Basic Structure

```
#include <iostream>
#include <cstdlib>
using namespace std;
// Hello World Program
int main() {
    cout << "Hello World\n";
    exit(1);
    cout << "Hello Heaven\n";
    return 0;
}
```

- return / exit : terminate an application immediately.
- If there are several return/exit, the program ends at the first return/exit.

- Terminate here
- Output is just

```
Hello World
```

Output Operator: <<

```
cout << exp1 << exp2 << … << expn;
```

- cout: the output stream
  - Directing values to our screen
- exp (expression): value displaying on a screen
  - Numbers (integers and real numbers), characters, or texts
  - Raw values (1,2,3,…) or variable values (x,y,z,…)
  - Special characters: space, new line, tab

Output Operator: <<

- Show exp from left to right
- << doesn't add space, just connect values
  - You need to manually add space

endl = end line

Code:

```
cout << 2 << "Hello" << "World" << endl;
cout << 2 << " " << "Hello" << " " << "World" << endl;
cout << 2 << " " << "Hello World" << endl;
```

Output:

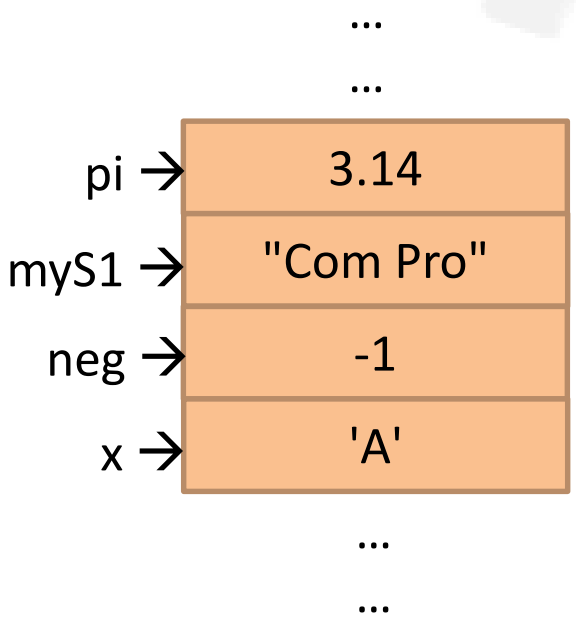
```
2HelloWorld
2 Hello World
2 Hello World
```

Basic Data Types

- Text
  - Character: char 'a'
  - Word/Phrase: string "abcdefg"
- Numbers
  - Integer: char, short, int, long
  - Real numbers: double, float
  - Signed vs. unsigned
    - Signed: spare 1 bit for +/-
    - Unsigned: always a positive number
- Boolean
  - true (non-zero) or false (0)

Variable

- A symbol or a name referred to a value stored in the memory
  - A "box" of memory stores a value (any data type)
  - A variable => 1 box of value in the memory



Variable Name

- Begin with an alphabet (a-z, A-Z) or an underscore (_)
  - Followed by alphanumeric character(s) (a-z, A-Z, 0-9) or an underscore (_)

☑ X, x, x2, G2000, _myclass, helLo_2worlD

☒ 2xyz, 3000, 44aa

Reserved Words / Identifiers

- Reserved words
  - Words with special meaning in C++
  - Can't be used as a variable name
  - E.g. *include, return, endl, if, switch, while*
- Identifiers
  - Names used to declare variables, functions, data types, etc
  - E.g. *main* (function), *n* (variable), *cout* (std variable), *int* (data type), *string* (data type)

Variable declaration

```
type varName;
```

Code:

```
int x;
int a, b;
unsigned int x1;
short z;
unsigned short uz;
long y;
unsigned long uy;
```

Code:

```
double c, d;
double mySum2;
unsigned double m;
float ee, sum;
unsigned float uFmoney;
char ch;
string str, myString;
bool isDigit;
```

Variable declaration

- Must be declared before use

Right:

```
int main() {
  int x, y;
  x = 1;
  y = x + 1;
  int z = 5;
  cout << x << y << z;
  return 0;
}
```

Wrong:

```
int main() {
  x = 1;
  y = x + 1;
  int z = 5;
  cout << x << y << z;
  int x, y;
  return 0;
}
```

Variable Assignment

- Assign a value to a variable

```
varName = exp;
```

Code:

varName on the left of =

```
int main() {
   int a,b;
   double x;
   a = 3;
   b = 4;
   x = 68.9546;
}
```

Value on the right of =

Variable Initialization

- Assign a value to a variable when declared

Code:

```
int main() {
   int a = 3;
   int b = 4, c = 5;
   double x = 68.9546;
}
```

Use comma (,) if you want to initialize several variables (same type) on the same line

Character

- An alphabet, a number, or a symbol
- Enclosed within a pair of single quotes ('')
- E.g. 'A', 'b', '9', '+'
- Non-printable characters:
  - Newline: '\n'
  - Tab: '\t'
  - Return: '\r'

String Literal

- A series of characters
- Enclosed within a pair of double quotes ("")
  - E.g. "Hello", "World", " ", "12345"
- Standard C++ defines its string in <cstring>

String Declaration and Initialization

```
string s1;
string s2 = "New York";
string s3(60, '*');
string s4 = s3;
string s5(s2, 4, 2);
```

Not initialized = empty string ""

Output if cout:

```
s1 =
s2 = New York
s3 = ************************************************************
s4 = ************************************************************
s5 = Yo
```

String Concatenation

```
#include <cstring>
int main(){
        string firstname = "John";
        string lastname = "Doe";
        string fullname = firstname + " " + lastname;
        cout << "Your full name is " << fullname << endl;
        return 0;
}
```

Output :

Length of A String

- #include <cstring>
- Function: strlen("string")
- Special character's length = 1
- Space's length = 1

Code:

Length = 1

Length = 1

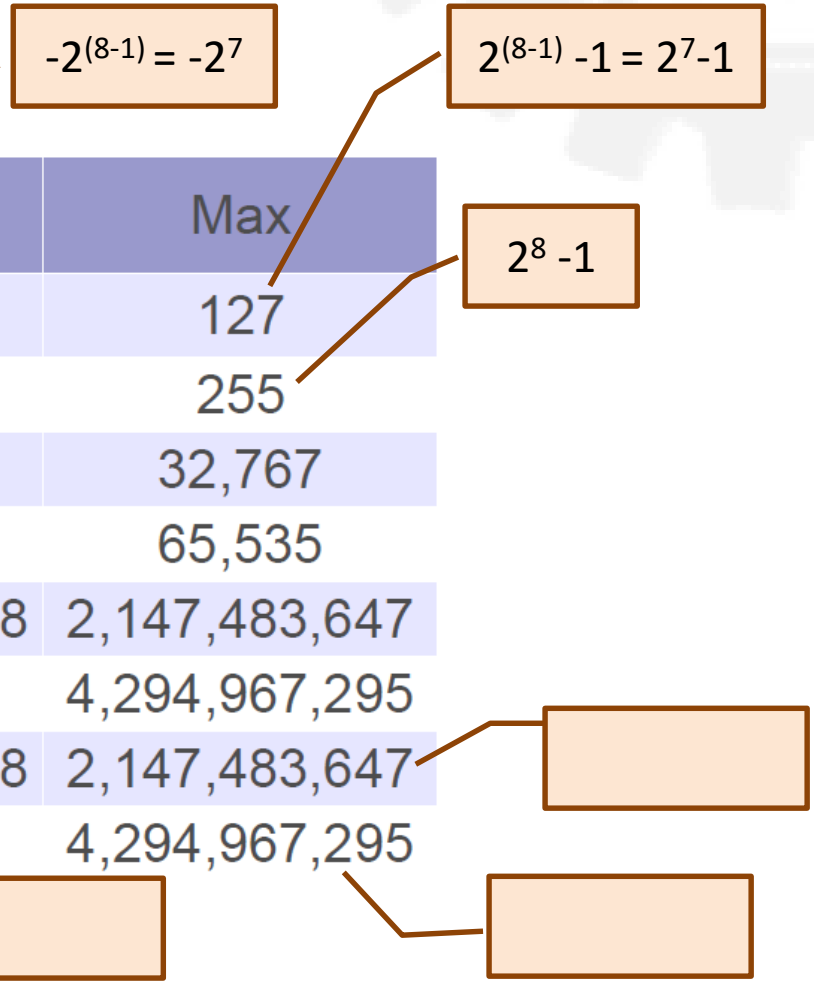
```
cout << strlen("Hello, World.\n") << '\n';
cout << strlen("Hello, World.") << '\n';
cout << strlen("Hello, ") << '\n';
cout << strlen("H") << '\n';
cout << strlen("") << '\n';
```

Output:

Integers

- For 32-bit Architecture

Types	Bits	Min	Max
char	8	-128	127
unsigned char	8	0	255
short	16	-32,768	32,767
unsigned short	16	0	65,535
int	32	-2,147,483,648	2,147,483,647
unsigned int	32	0	4,294,967,295
long	32	-2,147,483,648	2,147,483,647
unsigned long	32	0	4,294,967,295



Size of Data Type

- Function: sizeof( ) gives data size in byte

Code:

```
int main(){
        cout << "Size of char: " << sizeof(char) << endl;
        cout << "Size of short: " << sizeof(short) << endl;
        cout << "Size of int: " << sizeof(int) << endl;
        char a;
        int b;
        cout << "Size of a: " << sizeof(a) << endl;
        cout << "Size of b: " << sizeof(b) << endl;
        return 0;
}
```

Size of Data Type

- Function: sizeof( ) gives data size in byte

Output:

Arithmetic Operators

Operator	Meaning	Example
++/--	Pre	++n
- Negate	-n	
* Multiply	m * n	
/ Divide	m / n	
%	Remainder	m % n
+ Add	m + n	
– Subtract	m – n	Lower
++/--	Post	n++

1st

2nd

3rd

Last

If operators in the same level, compute from left to right

Use parentheses ( ) to change the order

Arithmetic Op Example

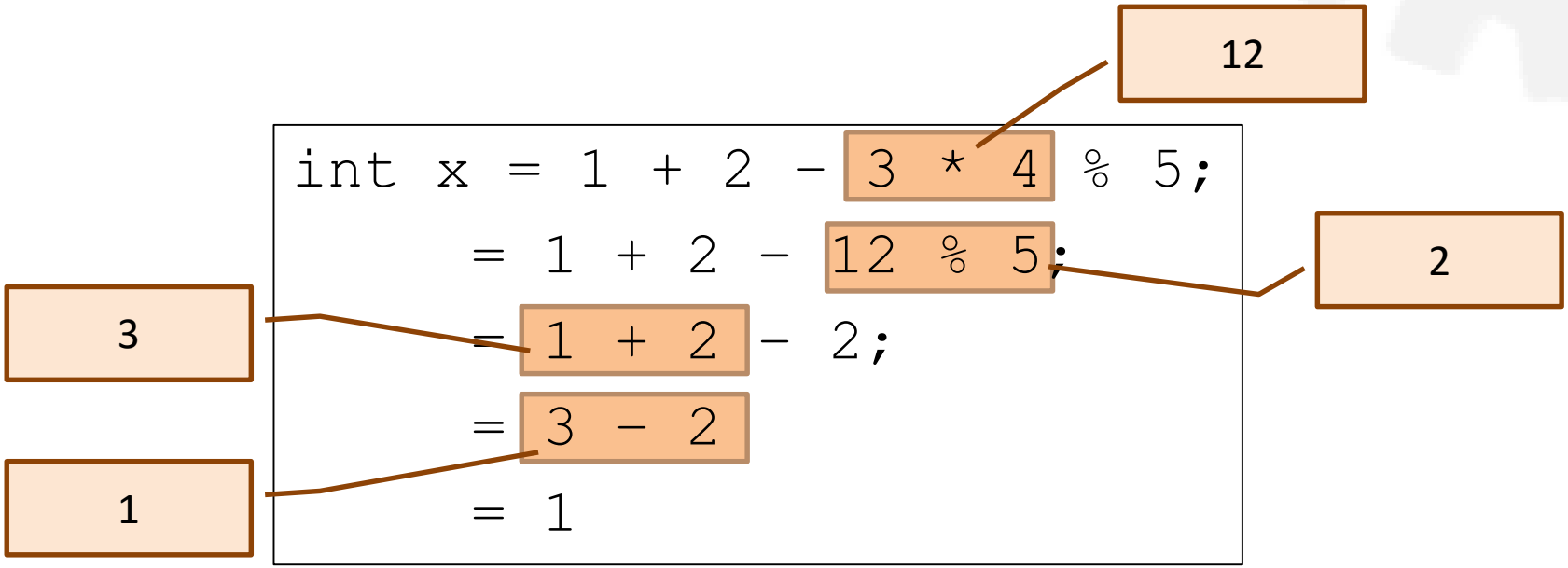
Code:

```
int main() {
   int a = 38, b = 5;
   cout << a << " + " << b << " = " << (a+b) << endl;
   cout << a << " - " << b << " = " << (a-b) << endl;
   cout <<      " - " << b << " = " << (-b) << endl;
   cout << a << " * " << b << " = " << (a*b) << endl;
   cout << a << " / " << b << " = " << (a/b) << endl;
   cout << a << " % " << b << " = " << (a%b) << endl;
   return 0;
}
```

Arithmetic Op Example

Output:

Precedence of Operator Example



Increment (++)

```
varName++;
++Varname;
varName = varName + 1;
```

- ++ operator increases *an integer* by one
  - Same as m = m + 1;
  - Pre-increment: ++m
    - Increase by 1 -> do the other operation
  - Post-increment: m++
    - Do the other operation -> increase by 1

++ Example

Code:

```
int main() {
   int a = 66, b;
   b = ++a;
   cout << "a = " << a << ", b = "
   << b << endl;

   b = a++;
   cout << "a = " << a << ", b = "
   << b << endl;

   cout << "a = " << a++ << endl;
   cout << "a = " << a << endl;
   cout << "a = " << ++a << endl;
   return 0;
}
```

Output:

Decrement (--)

```
varName--;
--Varname;
varName = varName - 1;
```

- -- operator decreases *an integer* by one
  - Same as m = m - 1;
  - Pre-decrement: --m
    - Decrease by 1 -> do the other operation
  - Post-decrement: m--
    - Do the other operation -> decrease by 1

-- Example

Code:

```
int main() {
   int a = 66, b;
   b = --a;
   cout << "a = " << a << ", b = "
   << b << endl;

   b = a--;
   cout << "a = " << a << ", b = "
   << b << endl;

   cout << "a = " << b-- << endl;
   cout << "a = " << b << endl;
   cout << "a = " << --b << endl;
   return 0;
}
```

Output:

Composite Assignment

```
varName op= exp;                //(1)
varName = varName op exp;       //(2)
```

New value of varName after operating with exp

Current value of varName

Code #1:

```
int main() {
   int n = 1;
   n += 8;
}
```

Code #2:

```
int main() {
   int n = 1;
   n = n + 8;
}
```

- Statement (1) and (2) give the same result
- op = + - * / %

Composite Assignment Example

Code:

```
int main() {
   int n = 44;
   n += 9;
   cout << n << endl;
   n -= 5;
   cout << n << endl;
   n *= 2;
   cout << n << endl;
   return 0;
}
```

Output:

Overflow and Underflow

- Overflow: value > max
- Underflow: value < min
- Divide by zero
- #include <climits>
  - SHRT_MAX = 32,767 (max value of short)
  - SHRT_MIN = -32,768 (min value of short)

Overflow Example

Code:

```
#include <climits>
int main() {
   short n = SHRT_MAX-1;
   cout << n++ << endl;
   cout << n++ << endl;
   cout << n++ << endl;
   cout << n++ << endl;
   return 0;
}
```

Output:

Underflow Example

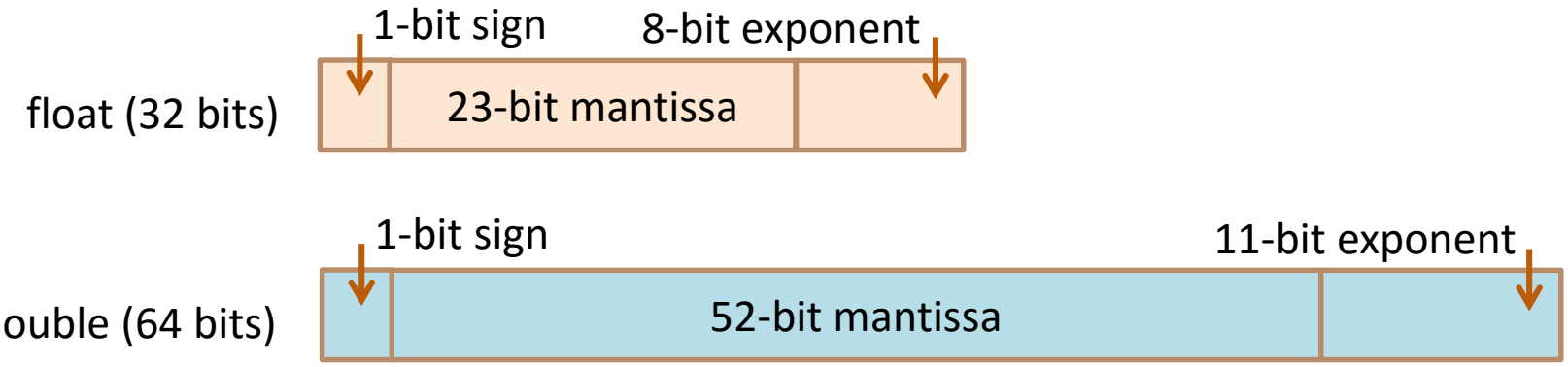
Code:

```
#include <climits>
int main() {
   short n = SHRT_MIN+1;
   cout << n-- << endl;
   cout << n-- << endl;
   cout << n-- << endl;
   cout << n-- << endl;
   return 0;
}
```

Output:

Real Numbers

- Can have decimals
- 3 Types: float (32 bits), double (64 bits), long double (64, 80, 96, or 128 bits)
- Floating-point arithmetic: + - * / (no %)
  - Division: does not truncate the result



Round-off Errors

- Might occur when computer does arithmetic on a rational number

Code:

```
int main() {
    double x = 1/3.0;
    double y = (x * 3.0) - 1.0;
    cout << "y = " << y << endl;
    return 0;
}
```

Output:

Precision Display

Code:

```
#include <iomanip>

int main() {
        double x = 1/3.0;
        double y = (x * 3.0) - 1.0;
        cout << "y = " << y << endl;

        cout << std::fixed;
        cout << "y = " << std::setprecision(5) << y << endl;
        cout << "y = " << std::setprecision(25) << y << endl;

        float a = 1/3.0;
        float b = (a * 3.0) - 1.0;
        cout << "b = " << std::setprecision(25) << b << endl;
        return 0;
}
```

Precision Display

Output:

Type Casting

Automatic Casting:

```
real op integer → real
```

Explicit Casting:

```
newType(varName)
(newType)varName
```

- Change a number from one type to another
  - char, short, int, long, float, double, long double
- op: + - * /

Type Casting Example (1)

Code:

```
int main() {
    float f = 2 * 3.14159 * 10;
    int n = 2 * 3.14159 * 10;
    float x = float(1)/3;

    cout << "f = " << f << endl;
    cout << "n = " << n << endl;
    cout << "x = " << x << endl;
    return 0;
}
```

Output:

Type Casting Example (2)

Code:

```
int main() {
    int x = int(1.0/5);
    int y = float(1)/5;
    float y2 = 1/5;
    float y3 = float(1/5);
    float y4 = float(1)/5;
    float y5 = 1.0/5;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "y2 = " << y2 << endl;
    cout << "y3 = " << y3 << endl;
    cout << "y4 = " << y4 << endl;
    cout << "y5 = " << y5 << endl;
    return 0; }
```

Output:

Character

- Hold 1 byte (8 bits)
- Can be considered as an integer
  - Value according to the ASCII code
- Arithmetic operations:

Code:

```
int main() {
    char c = 54;
    char d = 2 * c - 7;
    cout << c << " " << d << " ";
    c += d % 3;
    cout << c << endl;
    return 0; }
```

Output:

Manipulating Characters

Code:

```
int main() {
   char c = 64;
   cout << c++ << " ";
   cout << c++ << " ";
   cout << c++ << " ";
   cout << c++ << endl;
   c = 96;
   cout << c++ << " ";
   cout << c++ << " ";
   cout << c++ << " ";
   cout << c++ << endl;
   return 0;
}
```

Output:

Casting char as int

- Cast char as int if we want to display a numerical value

Code:

```
int main() {
    char c = 'A';
    cout << c++ << " " << int(c) << endl;
    cout << c++ << " " << int(c) << endl;
    cout << c++ << " " << int(c) << endl;
    return 0;
}
```

Output:

Boolean

- Declaration: bool a;
- Value: true or false
- true: non-zero
  - e.g. -5, -1, 1, 5, 9, ...
- false: zero
- Stored as 1 (true) or 0 (false)

Boolean Example

Code:

```
int main() {
    bool a = true, b = false;
    bool c = 3 < 4;
    bool d = 3 > 4;
    bool e = 0.4, f = 0;
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "c = " << c << endl;
    cout << "d = " << d << endl;
    cout << "e = " << e << endl;
    cout << "f = " << f << endl;
    return 0;
}
```

Output:

Alphanumerical bool Values

Code:

```
int main() {
	bool b = true;
	cout << std::boolalpha << b << '\n';
	cout << std::noboolalpha << b << '\n';
	return 0;
}
```

Output:

Constants

```
const type varName = value;
```

- CANNOT change the value of this object
  - Compilation error if trying to change the value
- Declare by putting const before an object type

Constant Example

Code:

```
int main() {
    const char beep = '\b';
    const float pi = 3.1415927;
    cout << "2 x pi = " << 2 * pi << beep << endl;
    return 0;
}
```

Output:

Input Operator: >>

```
cin >> var1 >> var2 >> … >> varn;
```

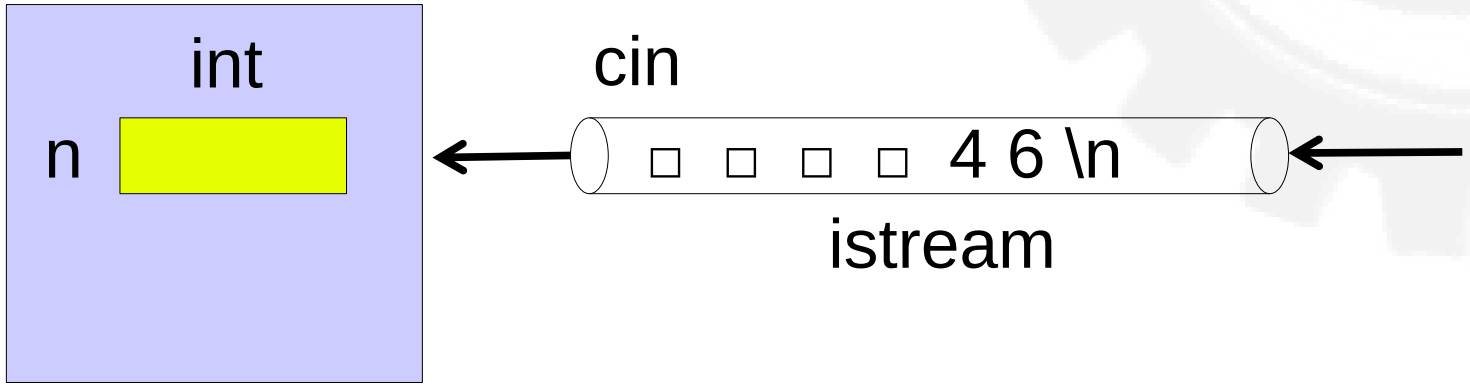
- cin: the input stream
  - Getting values from our keyboard (what has been typed to our screen)
- var: variable containing the input value
  - Declare var before used with cin

Formatted Input

- Input passes through an istream (input stream)
  - Defines behavior of cin
- The most common behavior is the use of the extraction or input operator >>
- Two operands:
  - The istream, which is extracting characters
  - The object to which it copies from those characters
- This process of forming a typed value from raw input character is called formatting.

The Extraction Operator

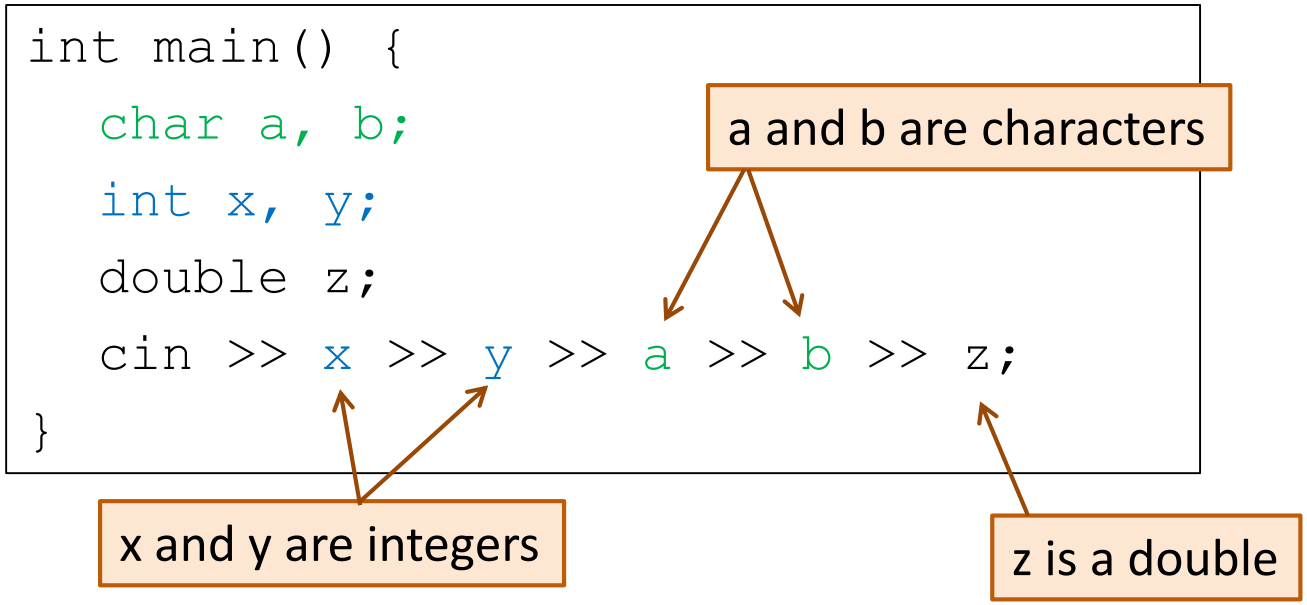
```
int n;
cin >> n;
```



- This input has 7 characters
  - 4 space characters, '4', '6', and '\n'
- We want to get `int`
  - So look for digits to form an integer
  - Extracts and ignore white spaces until it encounters a non-whitespace character
  - Stop when seeing a non-digit character

Input Operator: > >

- Read value depends on the type of var that you declared



Input Operator: > >

- User separate several input values by using a space or a new line

Code:

```
int main() {
   int a, b;
   double x;
   cout << "Enter 3 values: ";
   cin >> a >> b >> x;
   cout << a << " " << b << " "
   << x << endl;
}
```

Output #1:

```
Enter 3 values: 1 2 3.5
1 2 3.5
```

Output #2:

```
Enter 3 values: 1
2
3.5
1 2 3.5
```

What we enter

Input Operator Example (1)

Code:

```
int main() {
   int a, b;
   cin >> a >> b;
   cout << "a + b = " << a + b << endl;
    return 0;
}
```

Output:

Input Operator Example (2)

Code:

```
int main() {
   int x = 0;
   double y = 2.0;
   cout << "Enter int and double: ";
   cin >> x >> y;
   cout << "x + y = " << x+y << endl;
}
```

Output:

Take Home Message

- Basic C++ structure
- Output: cout << value; // value or variable
- Input: cin >> variable;
- Variables and types of variable
  - char, short, int, long, float, double, string, bool, constant
  - Know the differences (types and length)
  - Know how to declare and assign values to variables
- Number types
  - Arithmetic operations, overflow, underflow

References

- คณาจารย์คณะวิศวกรรมศาสตร์. (n.d.). **เอกสารชุดนำเสนอภาพและบรรยายวิชาการเขียนโปรแกรม (ส่วนกลาง)**
- คณาจารย์คณะวิศวกรรมศาสตร์. (n.d.). **เอกสารประกอบการสอนวิชาการเขียนโปรแกรม (ส่วนกลาง)**
- รศ. วิโรจน์ ทวีปวรเดช. (2554). **การเขียนโปรแกรมคอมพิวเตอร์ Computer Programming.** พิมพ์ครั้งที่ 2 โรงพิมพ์มหาวิทยาลัยขอนแก่น
- Cplusplus.com. (n.d.). **C++ Documentation**. สืบค้นเมื่อ 18 กุมภาพันธ์ 2555, http://www.cplusplus.com/
- Cplusplus.com. (n.d.). **Library Reference**. สืบค้นเมื่อ 18 กุมภาพันธ์ 2555, http://www.cplusplus.com/
- ISO/IEC 14882 Programming Language – C++