Unit 01
Introduction to basics of C++

ThanhNT

v1.0

Goals

- ✓ Share basics of C++
- ✓ Understand differences between C and C++
- ✓ Run some lines of C++ code

Outline

I. Getting started

II. C++ basics

1. Basic syntax
2. C versus C++
3. Input and output
4. Pointer
5. String
6. Reference variable
7. Namespace
8. Handling errors

For references

- learncpp.com
- tutorialspoint.com/cplusplus
- Google-sama

GETTING STARTED

Why?

- C++ is very popular
- Knew C, why not C++?
- New works, new opportunities

How?

- Any text editor and any C++ compiler
  - g++ -c
  - g++ -o
- Any completed C++ IDE

What?

- C++ was developed by Bjarne Stroustrup at Bell Labs, starting in 1979
- As an extension to C
- C++ adds many new features
- It is an object-oriented language

C++ BASICS

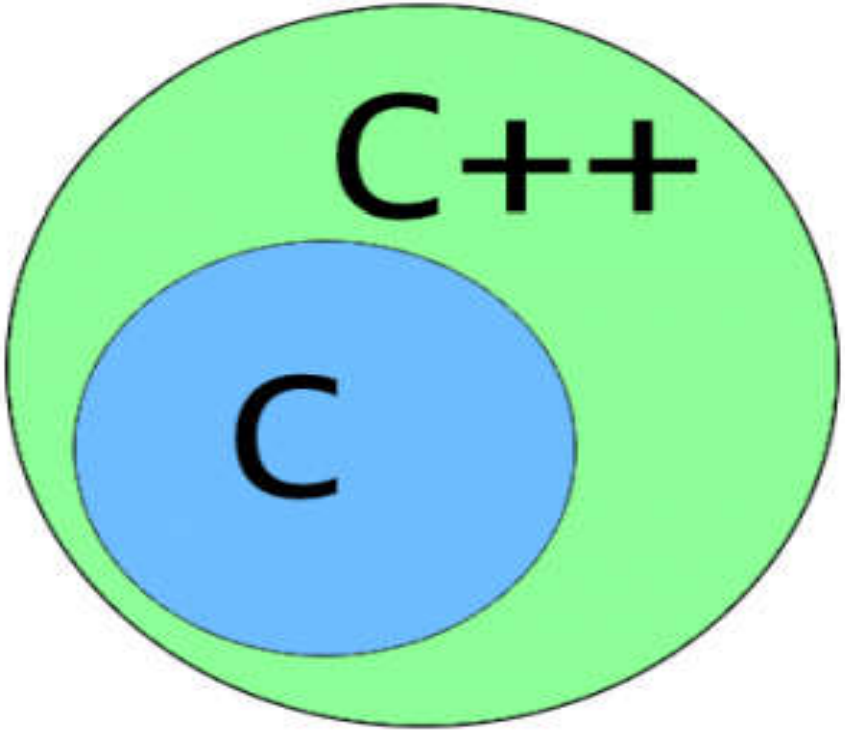
Basic Syntax

```
1   #include <iostream>
2
3   #define HELLO_WORLD "Hello world!"
4
5   int main()
6   {
7       std::string str = HELLO_WORLD;
8       std::cout << str << std::endl;
9       return 0;
10  }
11
```

```
Hello world!
[Finished in 1.4s]
```

C versus C++

C++ is a superset of C



C versus C++ (continue)

C	C++
C is a subset of C++	C++ is a superset of C
Procedural programming paradigm	Both procedural and object oriented programming paradigms
Function driven language	Object driven language
No polymorphism, encapsulation, and inheritance	Of course, yes
Data and functions are separate and free entities	Data and functions are encapsulated together in form of an object
No information hiding	Of course, yes

C versus C++ (continue)

C	C++
No function and operator overloading	Of course, yes
Not allow functions to be defined inside structures	Of course, yes
No namespace	Of course, yes
No reference variable	Of course, yes
No virtual and friend functions	Of course, yes
‘malloc’ and ‘free’	‘new’ and ‘delete’
No direct support for error handling	Of course, yes

Basic Input and output

- Why?
  - Only has to learn how to interact with the streams with many different kinds of devices
  - Left the details to the environment or operating system
- How?
  - C++ language provides input and output functionality through the C++ standard library
- What?

```
#include <iostream>

using namespace std;

int main(int argc, char const *argv[])
{
    char name[50];

    cout << "Please enter your name: ";
    cin >> name;
    cout << "Your name is: " << name << endl;

    return 1;
}

```

Pointer

- Why?
  - Pointer is typically seen as one of the most important parts of C/C++
- How?
  - Using operator ‘new’ and ‘delete’ to dynamically allocate single variables
- What?

```cpp
#include <iostream>

int main()
{
    int *ptr = new int; // dynamically allocate an integer
    *ptr = 7; // put a value in that memory location
    std::cout << *ptr << std::endl;
    delete ptr; // return the memory to the operating system. ptr is now a dangling pointer.
    std::cout << *ptr << std::endl; // Dereferencing a dangling pointer will cause undefined behavior
    delete ptr; // trying to deallocate the memory again will also lead to undefined behavior.
    return 0;
}
```

```
7
4009128
[Finished in 1.0s]
```

String

- Why?
  - This is C++, not C
  - String of C++ is easier, safer, and more flexible
- How?
  - C++ includes a built-in string data type as part of the standard library
- What?

```
1  #include <iostream>
2  #include <string>
3
4  int main()
5  {
6      std::string str = "This is a string";
7      std::cout << str << std::endl;
8      return 0;
9  }
```

```
This is a string
[Finished in 0.9s]
```

String (continue)

Function	Effect
Creation and destruction	
(constructor)	Create or copy a string
(destructor)	Destroy a string
Size and capacity	
capacity()	Returns the number of characters that can be held without reallocation
empty()	Returns a boolean indicating whether the string is empty
length(), size()	Returns the number of characters in string
max_size()	Returns the maximum string size that can be allocated
reserve()	Expand or shrink the capacity of the string
Element access	
[], at()	Accesses the character at a particular index
Modification	
=, assign()	Assigns a new value to the string
+=, append(), push_back()	Concatenates characters to end of the string
insert()	Inserts characters at an arbitrary index in string
clear()	Delete all characters in the string
erase()	Erase characters at an arbitrary index in string
replace()	Replace characters at an arbitrary index with other characters
resize()	Expand or shrink the string (truncates or adds characters at end of string)
swap()	Swaps the value of two strings
Input and Output	
>>, getline()	Reads values from the input stream into the string
<<	Writes string value to the output stream
c_str	Returns the contents of the string as a NULL-terminated C-style string
copy()	Copies contents (not NULL-terminated) to a character array
data()	Returns the contents of the string as a non-NULL-terminated character array

String (continue)

Function	Effect
	String comparison
==, !=	Compares whether two strings are equal/unequal (returns bool)
<, <=, > >=	Compares whether two strings are less than / greater than each other (returns bool)
compare()	Compares whether two strings are equal/unequal (returns -1, 0, or 1)
	Substrings and concatenation
+	Concatenates two strings
substr()	Returns a substring
	Searching
find	Find index of first character/substring
find_first_of	Find index of first character from a set of characters
find_first_not_of	Find index of first character not from a set of characters
find_last_of	Find index of last character from a set of characters
find_last_not_of	Find index of last character not from a set of characters
rfind	Find index of last character/substring
	Iterator and allocator support
begin(), end()	Forward-direction iterator support for beginning/end of string
get_allocator()	Returns the allocator
rbegin(), rend()	Reverse-direction iterator support for beginning/end of string

Reference variable

- Why?
  - Sometimes you don’t want to use pointer
- How?
  - C++ creates an alias to an initialized variable
  - Similar but still different to pointer
    - No NULL references
    - Can’t refer to another object
    - Must be initialized when it is created
- What?

```
5    int value = 5;
6    int &ref = value;
```

Namespace

- Why?
  - Resolve the conflict that two identifiers with the same name into the same scope
- How?
  - Define an area of code in which all identifiers are guaranteed to be unique
- What?

```
1   namespace foo
2   {
3       void doSomething()
4       {
5           // code
6       }
7   }
8   int main()
9   {
10      foo::doSomething();
11      return 0;
12  }
```

Handling errors

- Why?
  - When writing programs, there are always different kinds of errors
  - Errors should be commonly handled
- How?
  - Identify areas where assumptions may be violated
  - Write code that detects and handles any violation of those assumptions
- What?
  - Detecting assumption errors
  - Handling assumption errors
  - Assert and NDEBUG
  - Exceptions

Thanks!

Q&A