02. A Tour of C++: Types

Data Structure and Algorithms

Hanjun Kim

Compiler **O**ptimization **Re**search **Lab**

Yonsei University

The minimal C++ program

return type parameter list comment

```
int main() { }    // the minimal C++ program
```

function name function body

minimal.cpp

- The minimal C++ program
  - Defines a function called `main`
  - Does not take any argument
  - Does nothing
- `main`
  - Required function in a C++ program
- Return type
  - Specifies what kind of result, if any, the function will return to its caller
- Comments
  - Text after `//` or between `/*` and `*/`

Hello world

Instruct the compiler to include iostream library into the file

```
#include <iostream>

int main() {
  std::cout << "Hello, World!\n";     /* hello world */
}
```

Write 2nd argument into 1st

Comment

New line

Standard output stream

hello.cpp

- A program that writes `Hello, World!`
- `std::cout`
  - `std::` specifies that the name `cout` is to be found in the standard-library namespace

Hello world

```
#include <iostream>

using namespace std;

int main() {
   cout << "Hello, World!\n";
}
```

make names from std visible without std::

hello2.cpp

- A program that writes `Hello, World!`
- `using namespace std;`
  - makes `cout` be found in the standard-library namespace without `std::`

Built-in Data Types

- Type:
  - Defines a set of possible values and a set of operations

Type	Keyword	Set of Values	Literal Values	Operations
Boolean	bool	truth values	true false	and, or, not
Character	char	characters	‘a’, ‘4’, ‘$’, ‘\n’	compare
Integer	int, short, long	integers	0, 34, -5, 0x13	add, subtract, multiply, divide
Floating-point	float, double	floating-point numbers	1.2, 3.14, .4, -0.45, 1.2e3	add, subtract, multiply, divide

Types

- C++ provides a set of types
  - E.g. bool, char, int, double
  - Called “built-in types”
- C++ programmers can define new types
  - Called “user-defined types”
  - We'll get to that eventually
- The C++ standard library provides a set of types
  - E.g. string, vector, complex
  - Technically, these are user-defined types
    - they are built using only facilities available to every user

Type	Keyword	Set of Values	Literal Values	Operations
String	string	sequences of characters	“abcd”, “hello world”	concatenate

Standard library type (not built-in type)

Declaration and Assignment

- Variable: A name that refers to a value
- Declaration statement: associates a type with a variable
- Assignment statement: associates a value with a variable

```
int a, b;        ← Declaration statement
a = 1234;        ← Assignment statement
b = 56;          ← literal
int c = a + b;   ← Combined declaration and assignment statement
```

variable → b

Trace

- Table of variable values after each statement

```
int a, b;
a = 1234;
b = 56;
int c = a + b;
a = b;
b = c;
```

a	b	c
undefined	undefined	undefined
1234	undefined	undefined
1234	56	undefined
1234	56	1290
56	56	1290
56	1290	1290

Input and Output

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

int main() {
  cout << “What is your name?\n";
  string name;
  cin >> name;
  cout << "Hello, " << name << '\n';
}
```

hello3.cpp

- String data type is useful for program input and output
- `cin>>first_name;`
  - reads characters until a whitespace character is seen (“a word”)
  - White space: space, tab, newline, …

String concatenation

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

int main() {
  cout << "please enter your first and second names\n";
  string first;
  string second;
  cin >> first >> second;              // read two strings
  string name = first + ' ' + second; // concatenation
  cout << "Hello, "<< name << '\n';
}
```

hello4.cpp

- `string name = first + ' ' + second;`
  - concatenate strings separated by a space

Integer

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

int main() {
  cout << "please enter your name and age\n";
  string name;
  int age;
  cin >> name >> age;     // read a string and an integer
  cout << "Hello, "<< name << " age: " << age << '\n';
}
```

hello5.cpp

- `cin >> name >> age;`
  - `cin` can read multiple words
  - `cin` can read integers directly without type conversion

String vs. Integer vs. Float

Operation	String	Integer	Float
cin >>	Reads a word	Reads a number	Reads a number
cout <<	Writes	Writes	Writes
+	Concatenates	Adds	Adds
+=x	Adds x at end	Increments by x	Increments by x
++	Error	Increments by 1	Increments by 1
-	Error	Subtracts	Subtracts
*	Error	Multiplies	Multiplies
/	Error	Divides	Divides
%	Error	Remainder	Error

Boolean

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

int main() {
  cout << "please enter a year\n";
  int year;
  bool isLeapYear;
  cin >> year;    // read the year
  // divisible by 4 but not 100
  isLeapYear = (year % 4 == 0) && (year % 100 != 0);
  // or divisible by 400
  isLeapYear = isLeapYear || (year % 400 == 0);

  if (isLeapYear) cout << year << " is a leap year\n";
  else cout << year << " is not a leap year\n";
}
```

leapyear.cpp

- Useful to control logic and flow of a program

C++14 hint

- You can use the type of an initializer as the type of a variable
  - // “auto” means “the type of the initializer”
  - auto x = 1; // 1 is an int, so x is an int
  - auto y = ′c′; // ′c′ is a char, so y is a char
  - auto d = 1.2; // 1.2 is a double, so d is a double
  - auto s = ″Howdy″; // ″Howdy″ is a string literal of type const char[]
    // so don’t do that until you know what it means!
  - auto sq = sqrt(2); // sq is the right type for the result of sqrt(2)
    // and you don’t have to remember what that is
  - auto duh; // error: no initializer for auto

Constants

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

const double pi = 3.14159;
int main ()
{
  double r=5.0;                // radius
  double circle;

  circle = 2 * pi * r;
  cout << circle << '\n';
}
```

const.cpp

- `const`
  - A notion of immutability
  - Programmers promise not to change this value
  - Data can be passed to functions without fear of it being modified
  - Used primarily to specify interfaces (The compiler enforces it)