Lecture
11_1.1

Introduction to C++

Alessandro Savino
Politecnico di Torino (Italy)
alessandro.savino@polito.it
www.testgroup.polito.it

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

– **This lecture presents a global overview of C++ programming language**

Prerequisites

– **Basic knowledge of C programming language**

Homework

– **Lab 0**

Outline

- History
- Character Set
- Tokens
- Keywords
- Identifiers
- Literals
- Punctuators
- Comments
- Compiler

History

- In 1980s *Bjarne Stroustrup* decided to extend the C language by adding some features from his favorite language *Simula 67.* Simula 67 was one of the earliest object oriented language. Bjarne Stroustrup called it “C with classes”.

- Later *Rick Mascitti* renamed as C++. Ever since its birth, C++ evolved to cope with problems encountered by users, and though discussions.

Standard Versions

- In 1985, the first edition of *The C++ Programming Language* was released, which became the definitive reference for the language. The first commercial implementation of C++ was released in the same year.

- In 1989 C++ 2.0 was released followed by the updated second edition of *The C++ Programming Language* in 1991. New features included multiple inheritance, abstract classes, static member functions, const member functions, and protected members.

Standard Versions

- In 1990, *The Annotated C++ Reference Manual* was published. This work became the basis for the future standard. Late feature additions included templates, exceptions, namespaces, new casts, and a boolean type.

- In 2011, C++11 was released which added more features and enlarged the standard library further (compared to it in 1998), providing more facilities for C++ programmers to use, with more additions during 2014 and planned for 2017.
  - some of them comes from the Boost libraries project!

Official Website

- On www.cplusplus.com you will find (almost) all you need, including:
  - Information
  - Tutorials
  - Reference
  - Articles
  - Forum

IDE

- **An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development.**
- **IDE normally consist of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.**
- **For lab activities in this course we will use CodeBlocks 13.12 stable release (www.codeblocks.org).**

C++ Character Set

- **Character set is a set of valid characters that a language can recognize. A character represents any letter, digit, or any other sign.**

Letters	A-Z, a-z
Digits	0-9
Special Symbols	Space + - * / ^ \ ( ) [ ] { } = != < > . ' " $, ; : % ! & ? _(underscore) # <= >= @
White Spaces	Blank spaces, Horizontal tab, Carriage return, New line, Form feed.
Other Characters	C++ can process any of the 256 ASCII characters as data or as literals

Tokens

- **The smallest individual unit in a program is known as a Token or lexical unit.**

- **Types of Tokens**
  1. **Keywords**
  2. **Identifiers**
  3. **Literals**
  4. **Punctuators**
  5. **Operators**

Keywords

- **Keywords are the words that convey a special meaning to the language compiler. These are reserved for special purpose and must not be used as normal identifier names.**

Keywords

asm continue float new signed try auto
default for operator sizeof typedef break
delete friend private static union case do goto
protected struct unsigned catch double if
public switch virtual char else inline register
template void class enum int return this volatile
const extern long short throw while

Identifiers

- Identifiers are names of the program given by user.

- Rules to write identifiers
  - Do not start with digits
  - No special symbols except _ (underscore)
  - No spaces

Literals

- Literals (constants) are data items that never change their value during a program run.

- Types of Literals:
  - Integer constants (e.g., 1, 077, 0xff)
  - Floating constants (e.g., 3.1415)
  - Character constants (e.g., ‘a’)
  - String literals (e.g., “hello”)

Integer Constants

- Integer constants are whole numbers without any fractional part.

- Three types of Integer constants
  - Decimal Integer constant
  - Octal Integer constant (starts with 0)
  - Hexadecimal Integer constant (starts with 0x)

Character Constants

- Character constants are single characters enclosed in single quotes, as in ‘z’.

String Literals

- Multiple character constants are treated as string literals, as in “z” or “hello world!”.

Escape Sequences

\a	Audible sound
\b	back space
\f	Formfeed
\n	Newline or Linefeed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\’	single quote
\”	double quote
\?	Question mark
\on	Octal number
\xHn	Hexadecimal number
\0	Null

Punctuators

- **The following characters are used as punctuators:**
  `[] () {} , ; : * ... = #`

  - **`Brackets [ ]` indicate single and multidimensional array subscripts.**
  - **`Parenthesis ( )` indicate function calls and function parameters.**

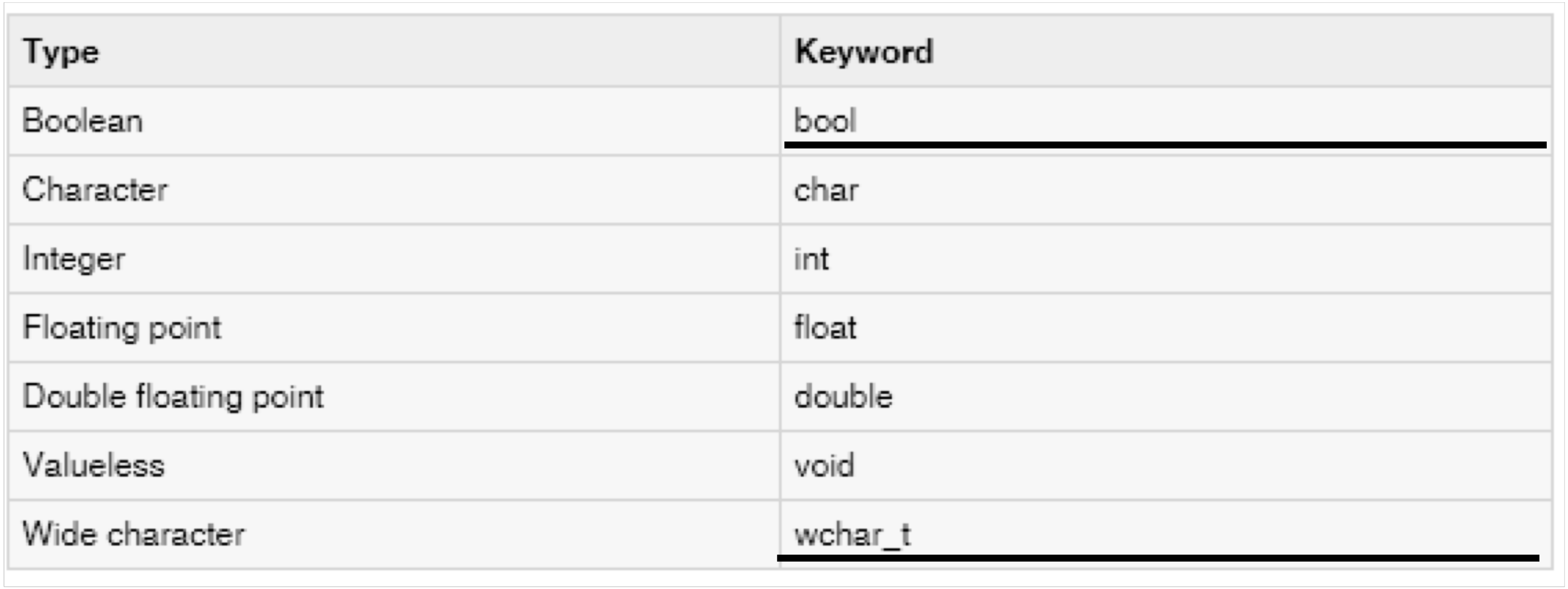
Punctuators

- `Braces {   }` indicate the start and end of a compound statement.
- `Comma ,` is used as separator in a function argument list.
- `Semicolon ;` is used as statement terminator.
- `Colon :` indicates a labeled statement.
- `Asterisk *` is used for pointer declaration.

Punctuators

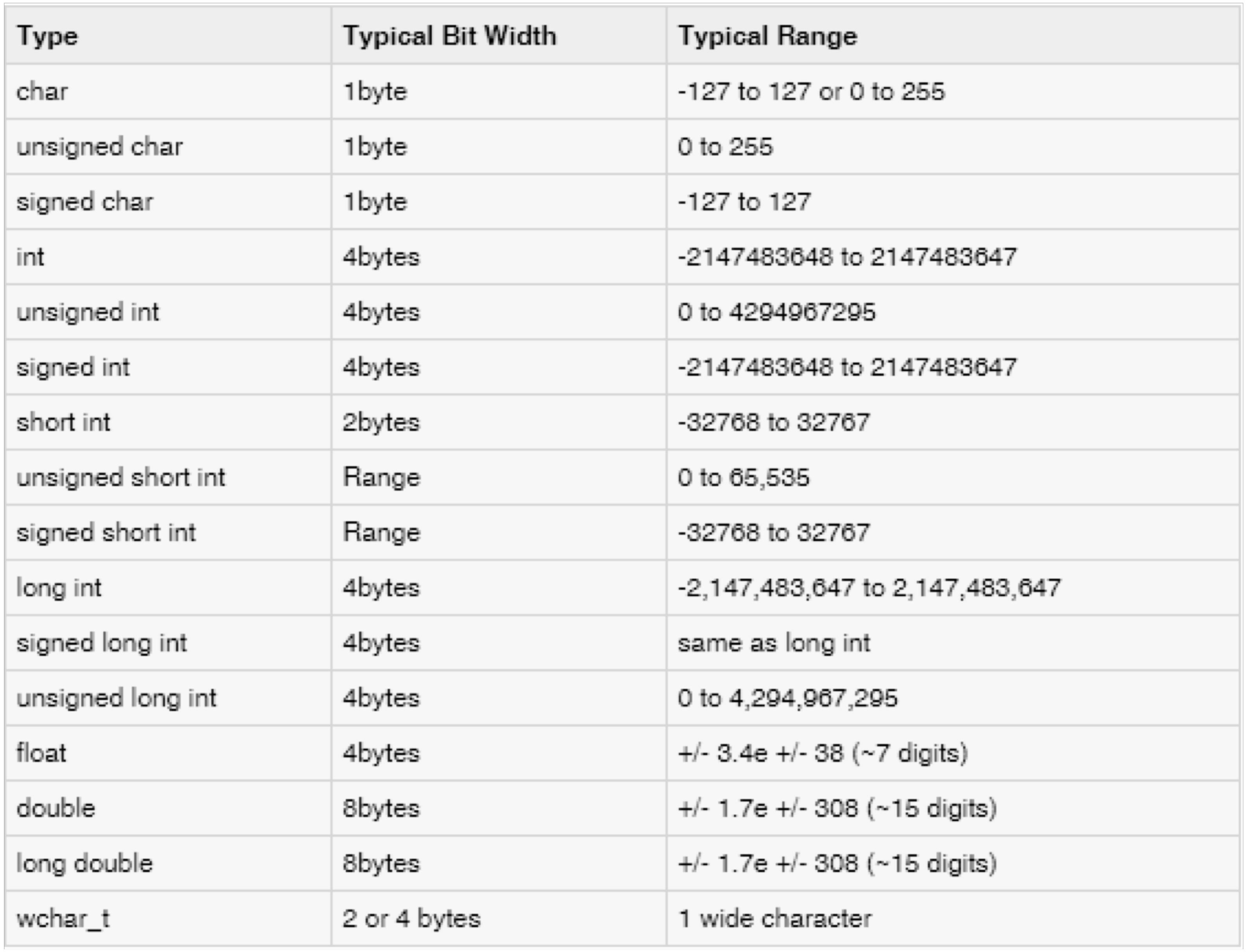
- `Ellipsis …` **are used in the formal argument lists of the function prototype to indicate a variable number of argument.**
  - **this works only if you have this cstdarg in your header**
    - **example: void function(int a, ...)**
- `Equal =` **is used for variable initialization and an assignment operator in expressions.**
  - **from C...**
- `Pound sign #` **is used for preprocessor directive.**
  - **do you remember #define?**

Data Types



Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Data Types



Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,647 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Operators

- Operators are tokens that trigger some computation when applied to variables and other objects in an expression.

- Types of operators
  - Unary operators (e.g., ~)
  - Binary operators (e.g., /)
  - Ternary operators (advanced, ?:)

Operators

- Operators are tokens that trigger some computation when applied to variables and other objects in an expression.

- Types of operators
  - Unary operators (e.g., ~)
  - Binary operators (e.g., /)
  - Ternary operators (advan

Pay attention to unary minus (int a=-5;) and binary minus (a=5-2;).

Some Unary Operators

&	Addresser operator
*	Indirection operator
+	Unary plus
-	Unary minus
~	Bitwise complement
++	increment operator
--	decrement operator
!	Logical negation

Some Binary Operators

+ - / *	**Arithmetic operators**
&& ||	**Logical operators**
< ! = >	**Relational operators**

Comments

- **Comments are pieces of codes that the compiler discards or ignores or simply does not execute.**

- **Types of comments:**
  - **Single line comments (//)**
  - **Multiline or block comments (/* … */)**

Role of the compiler

- A part of the compiler's job is to analyze the program code for correctness. If the meaning of the program is correct, then a compiler cannot detect errors.

- Types of errors:
  - Syntax Errors
  - Semantic Errors
  - Type Errors
  - Run-time Errors
  - Logical Errors

Role of the compiler

- **A part of the compiler's job is to analyze the program code for correctness. If the meaning of the program is correct, then a compiler cannot detect errors.**

- **Types of errors:**
  - **Syntax Errors**
  - **Semantic Errors**
  - **Type Errors**
  - **Run-time Errors**
  - **Logical Errors**

Pay attention!
Typically not detected by the compiler

Types of errors

- **Syntax Errors are occurred when rules of the program are wrong *i.e.,* when grammatical rule of C++ is violated.**

```
int a,b    (semicolon missing)
```

- **Semantic Errors are occur when statements not meaningful.**

```
x*y=z;
```

- **Type Errors are occurred when the data types are misused.**

```
int a=123.56;
```

Types of errors

- **Run-time Errors are occurred at the time of execution.**

- **Logical Errors are occurred when the logic of program is not proper.**

```
ctr=1;
while(ctr>10){ … }
```

Малые Автюхи, Калинковичский район, Республики Беларусь