Computer Science I

CS 135

1. Introduction to Programming

a. How to Develop a Program

b. Writing Pseudocode

c. First Elements of C++

- ✓ The basics of a C++ program
- ✓ Data types
- ✓ Arithmetic operators
- ✓ Expressions
- ✓ Variables
- ✓ Type casting
- ✓ ASCII characters
- ✓ Input and output

d. Looking Under the Hood

1.c First Elements of C++

The basics of a C++ program

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

int main()
{
      /* read data */
      cout << "Enter 3 numbers ";
      cin >> x >> y >> z;

      sum = x + y + z;    // calculate
      avg = sum/3.0;      // average

      cout << endl;
      cout << "The average is ";
      cout << avg << endl;

      return 0;
}
```

1.c First Elements of C++

The basics of a C++ program

➢ Structure of a C++ program

- ✓ a C++ program is a collection of one or more subprograms, called **functions**
- ✓ a subprogram or a function is a collection of statements that accomplishes something when executed
- ✓ every C++ program has at least one function called **`main`**
- ✓ the smallest individual unit of a program is called a “token”

➢ Tokens of a C++ program

- ✓ special symbols
- ✓ word symbols
- ✓ identifiers

1.c First Elements of C++

The basics of a C++ program

Special symbols

- ✓ mathematical symbols: `+` `-` `*` `/` ...
- ✓ punctuation marks: `.` `;` `?` `,` ...
- ✓ two-character symbols: `<=` `!=` `==` `>=` ...
- ✓ etc.

Word symbols

- ✓ `int`, `float`, `char`, `void`, `return`, etc.
- ✓ also called reserved words or **keywords**, as they belong to the language
  - cannot be redefined and reused
  - are always lowercase

1.c First Elements of C++

The basics of a C++ program

Identifiers

- **`main, x, y, sum, student_name,`** etc.
- identifiers are user-created names of things that appear in programs (variables, constants, functions, etc.):
  - consist of letters, digits, and the underscore character _
  - must begin with a letter or underscore
  - are case sensitive
- should be meaningful: **`student_name`** better than **`sdtnm`**
- there exists predefined identifiers, such as **`cout`** and **`cin`**
  - unlike reserved words, predefined identifiers may be redefined, but not a good idea as it can be confusing

1.c First Elements of C++

The basics of a C++ program

- Examples of legal and illegal identifiers

Identifier	Legal	Illegal	
`first name`		✗	(space break)
`first_name`	✓		
`firstName`	✓		
`1st_name`		✗	(first char is digit)
`first-name`		✗	(hyphen is symbol)
`_name1`	✓		

1.c First Elements of C++

The basics of a C++ program

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

int main()
{
        /* read data */
        cout << "Enter 3 numbers ";
        cin >> x >> y >> z;

        sum = x + y + z;     // calculate
        avg = sum/3.0;       // average

        cout << endl;
        cout << "The average is ";
        cout << avg << endl;

        return 0;
}
```

char symbols
keywords
identifiers
literals
strings
comments

1.c First Elements of C++

Data types

- Different data types for different programs
  - ✓ for example, some programs work with numbers (scientific calculation), some other programs manipulate names (alphabetizing lists) and some use both (grading)
  - ✓ numbers and words are distinct **data types**
- Categories of data types in C++
  - ✓ simple data type
    - integral data type = integer numbers (without a decimal)
    - floating-point data type = decimal numbers
    - enumeration data type = programmer-created type
  - ✓ structured data type & pointers

1.c First Elements of C++

Data types

- Integral data types
  - ✓ `int` represents integer numbers, for example:
    - `0`, `37`, `-45`, `12500` (no comma inside number)
    - minimum: `-2147483648`
    - maximum: `2147483647`
  - ✓ `char` represents any single character, for example:
    - `'a'`, `'2'`, `'B'`, `'$'`, `' '` (in single quotes)
    - also represents small integers between `-128` and `127`
  - ✓ `bool` represents logical (Boolean) values and can be only
    - `true` or `false` (these are keywords)

1.c First Elements of C++

Data types

- Floating-point data types
  - ✓ `float` represents real numbers, for example:
    - `75.924`, `-1.482`, `0.0018`, `180.00`
    - `7.5924e1`, `-1.482e0`, `1.8e-3`, `1.8e2` (in scientific notation)
    - maximum of 6 or 7 significant digits
    - bounds: `-3.4e38` and `3.4e38`
  - ✓ `double` also represent real numbers but with double precision (more significant digits and larger interval)
    - maximum of 15 significant digits
    - bounds: `-1.7e308` and `1.7e308`

1.c First Elements of C++

Arithmetic operators

- The C++ arithmetic operators are
  - ✓ addition `+`
  - ✓ subtraction `-`
  - ✓ multiplication `*`
  - ✓ division `/`
  - ✓ remainder `%` (mod operator: 11 % 3 = 2)
  - ✓ `+`, `-`, `*` and `/` can be used with both integral and floating-point data types
- Two types of operators
  - ✓ unary operators have only one operand: `-` ☐ or `+` ☐
  - ✓ binary operators have two operands: ☐ `*` ☐ or ☐ `+` ☐

1.c First Elements of C++

Arithmetic operators

- Order of precedence
  - ✓ unary operators are evaluated first
  - ✓ then all operations inside parentheses `(  )` are evaluated next
  - ✓ then `*`, `/` and `%` are at the same level of precedence and are evaluated next
  - ✓ finally `+` and `-` are at the same level of precedence and are evaluated last
  - ✓ when operators are on the same level, evaluation is performed from left to right
  - ✓ example: `3 + 7 * 6` evaluates to `45`
  - ✓ example: `(3 + 7) * 6` evaluates to `60`

1.c First Elements of C++

Expressions

- Integer expressions
  - ✓ all operands are integer
  - ✓ the result is an integer
  - ✓ example: `10 / 4` evaluates to `2`
  - ✓ example: `72 - 7 * (-60 % 8)` evaluates to `100`
- Floating-point expressions
  - ✓ all operands are floating-point
  - ✓ the result is floating-point
  - ✓ example: `10.0 / 4.0` evaluates to `2.25`
  - ✓ example: `72.36 - 0.09 * 4.0` evaluates to `72.0`

1.c First Elements of C++

Expressions

- Mixed expressions
  - ✓ operands are of different data types, integer and floating-point
  - ✓ example: `5.4 * 2 - 13.6 + 21 / 6`
- Evaluation rules for binary operators
  - ✓ if both operands are integer, then result is integer
  - ✓ if both operands are floating-point, then result is floating-point
  - ✓ if one operand is integer and the other floating-point, then the integer is changed to floating-point and the result is floating-point
  - ✓ example: `5.4 * 2` `- 13.6 +` `21 / 6` yields `0.2`

`10.8` `3`