
Lecture 1: C++ Basics

Dragan Mirkovic
Department of Computer Science
University of Houston

D. Mirkovic, C++ Programming, Spring 2005

Announcements

- Today:
 - C++ Basics
 - Ch 1. In Irvine.
 - Introducing C++
 - Basic terminology
 - Classes and objects
 - Standardization of C++
 - C/C++ differences

D. Mirkovic, C++ Programming, Spring 2005

Introduction

- C++ is one of the most popular programming languages today
 - Industrial-strength language
 - Incorporates the efficiency of C with OO features
 - Well-suited for development of large complex software systems and libraries
 - Strongly typed language
 - Compiler performs strict type checking on variables and expressions
 - Object-oriented (OO) approach:
 - Focus on objects that make up an application problem
 - Program structure: relationship between objects
 - Process-oriented approach:
 - Program is organized as a hierarchy of tasks (procedures)
 - Procedural programming languages (C, Fortran, Pascal)

D. Mirkovic, C++ Programming, Spring 2005

History of C++

- C was developed at Bell Labs around 1969-1973 for implementation of the UNIX operating system, by Dennis Ritchie.
- 90% of UNIX was then written in C.
- 1989 - ANSI Standard for C
- C++ was written by Bjarne Stroustrup at Bell Labs during 1983-1985.
- C++ is an extension of C. Prior to 1983, Bjarne Stroustrup added features to C and formed what he called "C with Classes".
- He had combined the use of classes and object-oriented features with the power and efficiency of C. The term C++ was first used in 1983.

D. Mirkovic, C++ Programming, Spring 2005

Objects

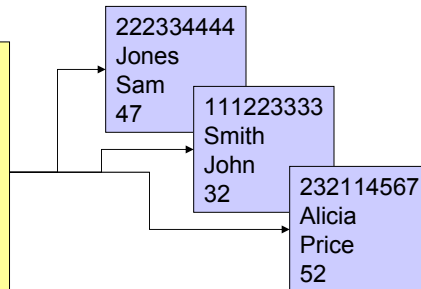
- Individual entities that comprise an application
- Examples
 - **Student**: A college student object
 - **Course**: A single course taken by a student
- Object characteristics:
 - **Attributes** (contents of an object)
 - **Operations** (functions of an object)
 - Object *encapsulates* both attributes and operations
- Example:
 - A **Student** object may have **Name** and **ID** as attributes and **Display** and **CalculateGrade** as operations

D. Mirkovic, C++ Programming, Spring 2005

Classes

- Description of the characteristics shared by all objects of the same type
 - Similar to *Abstract data type* (ADT) in C + inheritance, polymorphisms and much more
 - Object = An **instance** of the class
- Example:

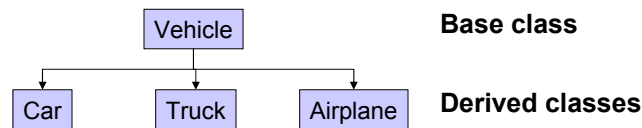
```
Class Student {  
    long id;  
    char lastName[30];  
    char firstName[30];  
    int totCredits;  
    void Input();  
    void Output();  
    float CalculateGrade();  
}
```



D. Mirkovic, C++ Programming, Spring 2005

Inheritance

- Different objects can have some common attributes and operations
- In some cases it is useful to define a **base class** which contains the common characteristics
- Corresponds to “Is a” relationship in Entity-relationship data model
- Example:



D. Mirkovic, C++ Programming, Spring 2005

Polymorphism

- A single name can denote objects of different types
- The actual type of the object doesn't have to be known at compile time
- Most programming languages do not support polymorphism
- Unions in C allow for a specific form of polymorphism
- In C++ polymorphic objects must be instances of classes related by inheritance
- Example: `calcSpeed()` function for `Vehicle-Car-Truck-Airplane` family
 - Dynamic binding vs. static binding

D. Mirkovic, C++ Programming, Spring 2005

Structure of a C++ program

- Example:

```
// my first program in C++  
  
#include <iostream.h>  
  
int main ()  
{  
    cout << "Hello World!";  
    return 0;  
}
```

- Example:

```
/* Equivalent program in C */  
  
#include <stdio.h>  
  
int main ()  
{  
    printf("Hello, World\n");  
    return 0;  
}
```

- Structure

- Comments (`//` line comment `/*` block comment `*/`)
- Preprocessor directives (`#`)
- the **main** function declaration.
- standard output stream **cout**, equivalent to **stdout**

D. Mirkovic, C++ Programming, Spring 2005

Variables, data types, constants.

- **Variable** = a portion of memory to store a determined value.
 - Each variable needs an identifier that distinguishes it from the others
`a = 5; b = 2; a = a + 1; result = a - b;`
- **Identifier** = a sequence of one or more letters, digits or underline symbols (`_`).
 - The length of an identifier is not limited
 - Variable identifiers should always begin with a letter or an underline character (`_`)
 - they cannot match any **key word**

D. Mirkovic, C++ Programming, Spring 2005

Key words in C++

- ANSI-C++ standard keywords:
`asm, auto, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double, dynamic_cast, else, enum, explicit, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t`
 - There may be additional reserved words depending on the compiler and operating system
 - The C++ language is "case sensitive"
-

D. Mirkovic, C++ Programming, Spring 2005

Data types

Type	Description	Bytes	Range
char	character or integer 8 bits length.	1	signed: -128 to 127 unsigned: 0 to 255
short	integer 16 bits length.	2	signed: -32768 to 32767 unsigned: 0 to 65535
int	integer 32 bits length.	4	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long	integer 64/32 bits length.	8/4	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
float	Single precision floating point number.	4	3.4e +/- 38 (7 digits)
double	Double precision floating point number.	8	1.7e +/- 308 (15 digits)
long double	Extended precision floating point number.	10	1.2e +/- 4932 (19 digits)
bool	Boolean value (true or false)	1	true or false
wchar_t	Wide character.	2	0 to 65535

D. Mirkovic, C++ Programming, Spring 2005

Declaration of variables

- In order to use a variable in C++, we must first declare it
- The syntax:
`int a;`
`float mynumber;`
- Initialization of variables:
`type identifier = initial_value ;`
Or
`type identifier (initial_value) ;`
- Examples:
`int a = 0; int a(0);`

```
#include <iostream.h>

int main ()
{
    // declaring variables:
    int a, b;
    int result;
    // process:
    a = 5; b = 2; a = a + 1;
    result = a - b;
    // print out the result:
    cout << result;
    // terminate the
    program: return 0;
}
```

D. Mirkovic, C++ Programming, Spring 2005

Scope of variables

- In C++ we can declare variables anywhere in the source code (not in C)

```
#include <iostream.h>

int Integer;
char aCharacter;
char string [20];
unsigned int NumberOfSons;

main ()
{
    unsigned short Age;
    float ANumber, AnotherOne;

    cout << "Enter your age:"
    cin >> Age;
    ...
}
```

- Global variables
- Local variables
- External variables

D. Mirkovic, C++ Programming, Spring 2005

Constants

- **Literals:**
 - Integer, floats, characters and strings
 - Examples:
 - Integer:

```
75      // decimal
0113    // octal
0x4b    // hexadecimal
```
 - Floating point numbers:

```
3.14159 // 3.14159
6.02e23 // 6.02 x 1023
```
 - Character and strings:

```
'p'
"Hello world"
```
- **Defined constants**

```
#define PI 3.14159265
#define NEWLINE '\n'
#define WIDTH 100
```
- **Declared constants**
- With the **const** prefix you can declare constants with a specific type exactly as you would do with a variable:

```
const int width = 100;
const char tab = '\t';
```

D. Mirkovic, C++ Programming, Spring 2005

Operators

- **Assignment:** =
- **Arithmetic operators:** +, -, *, /, %
- **Compound arithmetic operators:**
+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=
- **Increment/Decrement operators:** ++, --
- **Relational operators:** ==, !=, >, <, >=, <=
 - According to the ANSI-C++ standard, the result is a `bool` value (**true** or **false**)
 - Different from C (`int`)
- **Logic operators:** !, &&, ||
- **Conditional operator:** ?
- **Bitwise Operators:** &, |, ^, ~, <<, >>
- **Explicit type casting Operators:** (type) or type()

D. Mirkovic, C++ Programming, Spring 2005

Storage Duration

- All named objects in C++ have a certain lifetime determined by its *storage class*
- *Automatic storage duration: (default)*
 - Object is destroyed after exiting the block
 - `auto` keyword
- *Static storage duration:*
 - The object has the same lifetime as the program
 - `static` keyword
- *Dynamic storage duration:*
 - `new` and `delete` keywords

D. Mirkovic, C++ Programming, Spring 2005

C/C++ Differences

- New keywords
- Single line comments (`//`)
- Variable declaration
 - Only at the beginning of their block in C
 - Anywhere in the block prior to their 1st use
- Cast operator (2 ways in C++)
- Function prototypes are required
- Type-Safe Linkage
- Named constants in C++

D. Mirkovic, C++ Programming, Spring 2005

Reference Parameters

- Passing arguments by reference in C++

```
void swap(int &x, int &y)
{
    int tmp = x;
    x = y;
    y = tmp;
}
int A = 20, B = 10;
swap(A, B);
cout << A << ', ' << B;
```

- Passing arguments by reference in C

```
void swap(int *x, int *y)
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}
int A = 20, B = 10;
swap(&A, &B);
printf("%d, %d\n", A, B);
```

D. Mirkovic, C++ Programming, Spring 2005

Stream I/O

- Stream Output
 - The stream output operator (<<) appends an expression to the output stream
 - Examples:

```
cout << 10;           // prints number 10 on screen
cout << "Hello";     // prints Hello on screen
cout << x;           // prints the content of x variable on screen
cout << "Hello, I am " << age << " years old.";
```
 - When writing an expression use () to force evaluation of the expression before printing

```
cout << (10 + x);
```
 - Use a newline character '\n' to force line breaks

```
cout << (10 + x) << '\n';
```
 - Use endl stream manipulator a newline and flush the output buffer

```
cout << (10 + x) << '\n';
```

D. Mirkovic, C++ Programming, Spring 2005

Stream I/O

- Stream Input

- The stream input operator (>>) extracts data from an input stream

- Examples:

```
int n;
cin >> n; //read from
input stream
```

- By default operator skips whitespace (tabs, spaces, new lines)

```
int a, b; float r; char
ch;
cin >> a >> b >> r >> ch;
```

- `cin.get()` can be used for character input

- Example:

```
#include <iostream.h>
char name[80], ch = '\0';
int i = 0;
cout << "Enter your name:"
while(1)
{
    cin.get(ch);
    if(ch=='\n') break;
    name[i++] = ch;
}
name[i] = '\0';
```

D. Mirkovic, C++ Programming, Spring 2005

Summary

- C++ is an industrial-strength language

- Incorporates the efficiency of C with OO features
- Well-suited for development of large complex software systems and libraries
- Strongly typed language
- Basic terminology:
 - Objects, Classes, Inheritance, Polymorphism
- Differences between C and C++
- Stream I/O

D. Mirkovic, C++ Programming, Spring 2005

Homework

- Exercises 1-4, page 20 in text.