

EL2310 – Scientific Programming

Lecture 13: Intro to C++



Andrzej Pronobis
(pronobis@kth.se)

Royal Institute of Technology – KTH

Overview

Lecture 13: Intro to C++

- Differences between C and C++

- Namespaces

- Printing and User Input

- References and Pointers

- Allocating Memory Dynamically

Introduction to Object Oriented Paradigm



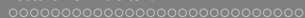
Course so far

- ▶ **MATLAB**: Using program to achieve a goal
- ▶ **C**: Learning how to program



Rest of the course

- ▶ C++
 - ▷ Writing extendable programs in C++
 - ▷ Object Oriented Programming
 - ▷ Using other peoples code
 - ▷ Extending other peoples code
 - ▷ Writing re-usable code



Today

- ▶ Intro to C++
- ▶ Intro to OOP



C++

- ▶ Developed by Bjarne Stroustrup starting from 1979 at Bell Labs
- ▶ Adds object oriented features (e.g. classes) to C
- ▶ C with Classes, renamed to C++ (guess why? :-)
- ▶ Influenced many other languages: C#, Java
- ▶ Objective-C use a different approach to adding classes to C
- ▶ The C++ standard library incorporates:
 - ▷ The C standard library with small modifications
 - ▷ STL (Standard Template Library)
- ▶ Constantly developed: C++11



Lecture 13: Intro to C++

Differences between C and C++

Namespaces

Printing and User Input

References and Pointers

Allocating Memory Dynamically

Introduction to Object Oriented Paradigm

C++ Compiler

- ▶ Use `g++` instead of `gcc`
- ▶ Usage and command line options are the same as for `gcc`
- ▶ Make sure you know how to use `make` for this part of the course!



File naming conventions

- ▶ We named files in C `.c` (source) and `.h` (header)
- ▶ In C++ the ending is typically `.cc` or `.cpp` for source files and `.h`, `.hh` or `.hpp` for header files
- ▶ In this course we will use `.cpp` and `.h`

Basic data types

- ▶ All data types from C can be used plus e.g.
- ▶ `bool`: **boolean value** `true/false`
- ▶ `string`: **“real” string** (use `#include <string>`)

Declaration of variables

- ▶ You no longer need to declare the variable at the beginning of the function (scope), as was the case for pre C99
- ▶ Useful rule of thumb: Declare variables close to where they're used.
- ▶ For instance:

```
for(int i=0; i<N; i++) {...}
```

i only defined within loop

- ▶ Use specific names for counters, e.g. *i*, *j*, *k*, ...

Lecture 13: Intro to C++

Differences between C and C++

Namespaces

Printing and User Input

References and Pointers

Allocating Memory Dynamically

Introduction to Object Oriented Paradigm

Namespaces

- ▶ In C all functions share a common `namespace`
- ▶ This means that there can only be one function for each function name
- ▶ In C++ can be placed in namespaces
- ▶ Syntax:

```
namespace NamespaceName {  
    void fcn(); ...  
}
```

Accessing functions in a namespace

- ▶ To access a function `fcn` in namespace `A`
`A::fcn`
- ▶ This way you can have more than one function with the same name but in different namespaces

using namespace

- ▶ Specifying the namespace all the time == a lot of typing

```
std::cout << "Apa" << std::endl;
```

- ▶ Extending a specific namespace,

- ▶ Ex.

```
using namespace std  
cout << "Apa" << endl;
```

- ▶ Avoid in header files

Task 1

- ▶ Write a program to test the idea of namespaces
- ▶ Define two functions `void fcn()`; inside namespaces `A` and `B`

Lecture 13: Intro to C++

Differences between C and C++

Namespaces

Printing and User Input

References and Pointers

Allocating Memory Dynamically

Introduction to Object Oriented Paradigm

Printing to Screen

- ▶ In C++ we use *streams* for input and output
- ▶ Output is handled with the stream `cout` and `cerr`
- ▶ All basic data types have the ability to add themselves to a stream for printing
- ▶ We use the `<<` operator
Ex: `cout << "Hello world";`
- ▶ To add a line feed use the “`\n`” as in C or the special `endl`
Ex: `cout << "Hello world" << endl;`

Printing to screen cont'd

- ▶ You can mix data types easily

- ▶ In C:

```
printf("The value is %d\n", value);
```

- ▶ In C++:

```
cout << "The value is " << value << endl;
```

- ▶ The stream `cerr` is the error stream

- ▶ Compare `stdout` and `stderr` in C

Formatting output

- ▶ Just like in C you can format the output in a stream
- ▶ You can use

- `width` number of characters for output to fill
 - `precision` number of digits
 - `fill` pad with a certain character

- ▶ **Syntax:**

```
cout.precision(4);  
cout.width(10);  
cout.fill('0');  
cout << 12.3456789 << endl;
```

- ▶ Will output 0000012.35
- ▶ Default precision=6, fill=' ' (space)

Getting input from the user

- ▶ Use streams also to get input from console
- ▶ Use the `cin` stream
- ▶ Ex:

```
int value;  
cin >> value;
```
- ▶ Using `cin` will flush the `cout` stream

Reading strings

- ▶ When reading with `cin` the input divided by spaces
- ▶ If you want to read an entire line, use `getline`
- ▶ Ex:

```
string line;  
getline(cin, line);  
cout << "The input was " << line << endl;
```


Hello world in C++

```
#include <iostream>
int main ()
{
    std::cout << "Hello World!";
    return 0;
}
```

- ▶ `<iostream>` replaced `<stdio.h>`
- ▶ Standard C++ header files are included without the suffix (no `.h` at the end)
- ▶ Here the `std` namespace is used, where `cout` is found

Task 2

- ▶ Write a program that reads the name and age of a person
- ▶ It should then print this info on the screen



Lecture 13: Intro to C++

Differences between C and C++

Namespaces

Printing and User Input

References and Pointers

Allocating Memory Dynamically

Introduction to Object Oriented Paradigm

References

- ▶ “Constrained” and “safer” pointers

- ▶ Compare

```
int a;
int *pa = &a;
int *pa = NULL;
*pa = 10;
int b;
pa = &b;
int *pc;
pc = pa;
```

```
int a;
int &ra = a;
-
ra = 10;    => a==10
int b;
-
-
-
```

Pointers vs References

- ▶ Try to use references when possible
- ▶ Much less error prone constructions
- ▶ References need to be assigned when constructed
- ▶ Ex: This is not allowed

```
int &x;  
int y;  
x = y; (assigned too late)
```

Passing Arguments by Reference

- ▶ Standard function calls are by value
- ▶ Value of the variable is copied into the function
- ▶ Pointers offered a way in C to do call by reference
- ▶ Call by reference avoids the need to copy all the data
- ▶ Ex: Not so good to copy an entire 10Mpixel image into a function, better to give a reference to it (i.e. tell where it is)
- ▶ In C++ we can use references

Passing Arguments by Reference, Cont'd

- ▶ Declaration: `void fcn(int &x);`
- ▶ Any changed to `x` inside `fcn` will affect the parameter used in the function call

- ▶ Ex:

```
void fcn(int &x)
{
    x = 42;
}
```

```
int main()
{
    int x = 1;
    fcn(x);
    cout << "x=" << x << endl;
}
```

- ▶ Will change value of `x` in the scope of `main` to 42

Lecture 13: Intro to C++

Differences between C and C++

Namespaces

Printing and User Input

References and Pointers

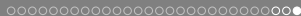
Allocating Memory Dynamically

Introduction to Object Oriented Paradigm

Dynamic Memory Allocation in C++

- ▶ In C++ the `new` and `delete` operators are used
- ▶ In C we used `malloc` and `free`
- ▶ Ex:

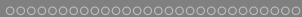
```
int *p = new int;  
*p = 42;  
...  
delete p;
```



new/delete for Arrays

- ▶ If you allocate an array with `new` you need to delete with `delete []`
- ▶ Ex:

```
int *p = new int[10];  
p[0] = 42;  
delete [] p;
```
- ▶ Typical mistake, forgotten `[]`



Lecture 13: Intro to C++

Differences between C and C++

Namespaces

Printing and User Input

References and Pointers

Allocating Memory Dynamically

Introduction to Object Oriented Paradigm



The Object-Oriented Paradigm

The motivation:

- ▶ We are trying to solve complex problems
 - ▷ Complex code with many functions and names
 - ▷ Difficult to keep track of all details
- ▶ How can we deal with the complexity?
 - ▷ Grouping related things
 - ▷ Abstracting things away
 - ▷ Creating hierarchies of things
- ▶ This also improves:
 - ▷ Code re-use
 - ▷ Reliability and debugging



Key Concepts of OOP

- ▶ Classes (types)
- ▶ Instances (objects)
- ▶ Methods
- ▶ Interfaces
- ▶ Access protection - information hiding
- ▶ Encapsulation
- ▶ Composition / aggregation
- ▶ Inheritance
- ▶ Polymorphism

Key Concepts of OOP

- ▶ Explaining the concepts on a car example

Next Time

- ▶ Object Oriented Programming
- ▶ C-project deadline Thursday 4th of October
- ▶ Friday is C exam
- ▶ New schedule for next week! More help sessions!