EL2310 – Scientific Programming

Lecture 15: C++

Carl Henrik Ek
(chek@csc.kth.se)

Royal Institute of Technology – KTH

Overview

Lecture 15: C++

- Wrap Up
- Inheritance continued
- Virtual Functions
- The Standard Template Library (STL)

Tasks

Wrap up of Course

Last time

- Function overloading
- Operator overloading

Today

- Inheritance
- virtual functions
- STL

Repetition

- **Namespaces**
- **Call by Reference**: &
- **Classes**: “extension” of structs
  - data, functions
  - constructor, destructor
- `this`, `const`

Inheritance

- Inheritance is a way to show a relation like “is a”
- Ex: A car is a vehicle
- A car inherits many of its properties from being a vehicle
- These same properties could also be inherited by a truck or a bus
- Syntax: `class Car : public Vehicle` to tell that Car inherits from Vehicle

Inheritance and Constructors

- If you have three classes A, B and C,
- where B inherits from A and C from B
- When you create C the constructor from the base classes (B and A) will be run first
- Execution order
  1. Constructor of A
  2. Constructor of B
  3. Constructor of C

Access specifiers

- `public:` can be accessed from outside class and from subclasses
- `private:` cannot be accessed from outside class or from subclasses
- `protected:` cannot be access from outside class but from subclasses

`virtual` functions

- To allow subclasses to re-define a function you declare it with the keyword `virtual`
- Ex:

```
class A {
public:
  virtual int print();
  ...
};
```

- This allows a subclass `B` to re-implement the `print` function

Polymorphism with `virtual` functions

- What function to run when dealing with `virtual` function is determined at run-time
- Depends on the objects
- If we have an object of type `A` we will use the function defined by `A`
- If we have an object of type `B` we will use the function defined by `B`

Pointers/references and `virtual` functions

- Assume `class B : public A`
- A pointer A *a; can be used to point to object of both type `A` and `B`
- Can only access the part of B that is inherited from A
- It will however look for alternative implementations of virtual functions
- References behave the same way

Subclasses as argument to function

- If a function want as argument a pointer/reference to an objects of type `A` it is ok to send in a pointer/reference to any subclass of `A`

Template

- C++ has a construction called template
- It offers a way to send data types as parameters
- Can have both template classes and functions
- Example of a template function:

```
template <class myType>
myType GetMax (myType a, myType b) {
  if(a>b){return a}
  else{return b}
}
```

- Example use: `GetMax<int>(4,5)` returns 5

Standard Template Library: STL

- Often want to use lists, vectors, etc.
- The Standard Template Library (STL) provides these
- Templates so that you can use any type of data in the lists, vectors, etc
- Examples:
  - `std::list<T>`
    Ex: `std::list<std::string> names;`
  - `std::vector<T>`
    Ex: `std::vector<double> values;`
  - `std::set<T>`
    Ex: `std::set<std::string> nameOfPerson;`
  - `std::map<T1,T2>`
    Ex: `std::map<int, std::string> nameOfMonth;`

Standard Template Library: STL

- Different structures are optimized towards different criteria, e.g.:
  - `std::list<T>`
    Cannot access elements with `x[i]`, need to use so called *iterators* to step through the list, can add/remove elements at low cost
  - `std::vector<T>`
    Can access elements with `x[i]`, typically with fixed sized vectors
  - `std::set<T>`
    Does not allow for redundant elements
  - `std::map<T1,T2>`
    Provides a mapping from one object to another
- They are also optimized to different manipulation times.
- They can be used in combination with `<algorithm>`s, like sort.

Often used: vector (example taken from C++ reference)

```
// erasing from vector
#include <iostream>
#include <vector>
using namespace std;

int main ()
{
  unsigned int i;
  vector<unsigned int> myvector;

  // set some values (from 1 to 10)
  for (i=1; i<=10; i++) myvector.push_back(i);
```

Often used: vector (example taken from C++ reference)

```
  // erase the 6th element
  myvector.erase(myvector.begin()+5);

  // erase the first 3 elements:
  myvector.erase(myvector.begin(),myvector.begin()+3);

  cout << "myvector contains:";
  for (i=0; i<myvector.size(); i++)
  cout << " " << myvector[i];
  cout << endl;

  return 0;
}
```

Standard Template Library: STL

- What would be suitable STL structures for:
  - a lottery drawing?
  - the people with their lottery numbers?
  - a library's book stocks?
  - a book stack?
- Differences are important:
  - How to insert an element?
  - How to access/find an element?
  - How to remove an element?

Standard Template Library: STL

- What would be suitable STL structures for:
  - a lottery drawing?
  - the people with their lottery numbers?
  - a library's book stocks?
  - a book stack?
- Differences are important:
  - How to insert an element?
  - How to access/find an element?
  - How to remove an element?

- A cricket team consists of a 11 players
- Players are either (“Rougly”)
  - Batsmen: **SR**, **Style**, **100:s**
  - Bowler: **Econ**, **5w**, **10w**
  - Wicket-keeper: **Ct**, **St**

Task

Create a cricket team of your own

- A team is a hierarchy,
  1. Team
  2. Player
  3. Role
- There are functionality that applies to all players,
  1. name
  2. statistics

MatLab - What you should have learned:

- Be comfortable working with MatLab
- Preparing scripts and functions using basic elements of programming (loops, branching, ...)
- Taking advantage of in-built functions (load data, plot data), especially the visualization capabilities.
- Translating a mathematical problem into MatLab code.

- Understand MatLab code if you see it.
- Know when (and how) to use MatLab in another course.

MatLab - What you should have learned:

- Be comfortable working with MatLab
- Preparing scripts and functions using basic elements of programming (loops, branching, ...)
- Taking advantage of in-built functions (load data, plot data), especially the visualization capabilities.
- Translating a mathematical problem into MatLab code.
- Understand MatLab code if you see it.
- Know when (and how) to use MatLab in another course.

MatLab - What you should have learned:

- Be comfortable working with MatLab
- Preparing scripts and functions using basic elements of programming (loops, branching, ...)
- Taking advantage of in-built functions (load data, plot data), especially the visualization capabilities.
- Translating a mathematical problem into MatLab code.

- Understand MatLab code if you see it.
- Know when (and how) to use MatLab in another course.

MATLAB - What you should have learned:

- Be comfortable working with MATLAB
- Preparing scripts and functions using basic elements of programming (loops, branching, ...)
- Taking advantage of in-built functions (load data, plot data), especially the visualization capabilities.
- Translating a mathematical problem into MATLAB code.

- Understand MATLAB code if you see it.
- Know when (and how) to use MATLAB in another course.

C - What you should have learned:

- Working with C: how to write, compile, link, execute.
- Declaring and initializing variables, basic data types, pointers(!), memory allocation(!)...
- Preparing programs using basic elements of programming (loops, branching, ...)
- Taking advantage of libraries (e.g. for printing data)
- Understand C code if you see it.
- Know when (and how) to use C in another course.

C - What you should have learned:

- Working with C: how to write, compile, link, execute.
- Declaring and initializing variables, basic data types, pointers(!), memory allocation(!)...
- Preparing programs using basic elements of programming (loops, branching, ...)
- Taking advantage of libraries (e.g. for printing data)
- Understand C code if you see it.
- Know when (and how) to use C in another course.

C - What you should have learned:

- Working with C: how to write, compile, link, execute.
- Declaring and initializing variables, basic data types, pointers(!), memory allocation(!)...
- Preparing programs using basic elements of programming (loops, branching, ...)
- Taking advantage of libraries (e.g. for printing data)
- Understand C code if you see it.
- Know when (and how) to use C in another course.

C - What you should have learned:

- Working with C: how to write, compile, link, execute.
- Declaring and initializing variables, basic data types, pointers(!), memory allocation(!)...
- Preparing programs using basic elements of programming (loops, branching, ...)
- Taking advantage of libraries (e.g. for printing data)
- Understand C code if you see it.
- Know when (and how) to use C in another course.

C++ - What you should have learned:

- What of C you can use in C++ and what C++ has to offer more (or in a different way) ...
- especially, the Object Oriented Programming Paradigm(!): Encapsulation, Polymorphism, Inheritance.
- Declaring classes and instantiating objects, accessing members, ...
- Understanding of 'conceptual programming', i.e. hiding of functions, declaring of static, const, virtual ...
- Understand C++ code if you see it.
- Know when (and how) to use C++ in another course.

C++ - What you should have learned:

- What of C you can use in C++ and what C++ has to offer more (or in a different way) ...
- especially, the Object Oriented Programming Paradigm(!): Encapsulation, Polymorphism, Inheritance.
- Declaring classes and instantiating objects, accessing members, ...
- Understanding of 'conceptual programming', i.e. hiding of functions, declaring of static, const, virtual ...
- Understand C++ code if you see it.
- Know when (and how) to use C++ in another course.

C++ - What you should have learned:

- What of C you can use in C++ and what C++ has to offer more (or in a different way) ...
- especially, the Object Oriented Programming Paradigm(!): Encapsulation, Polymorphism, Inheritance.
- Declaring classes and instantiating objects, accessing members, ...
- Understanding of 'conceptual programming', i.e. hiding of functions, declaring of static, const, virtual ...
- Understand C++ code if you see it.
- Know when (and how) to use C++ in another course.

C++ - What you should have learned:

- What of C you can use in C++ and what C++ has to offer more (or in a different way) ...
- especially, the Object Oriented Programming Paradigm(!): Encapsulation, Polymorphism, Inheritance.
- Declaring classes and instantiating objects, accessing members, ...
- Understanding of 'conceptual programming', i.e. hiding of functions, declaring of static, const, virtual ...
- Understand C++ code if you see it.
- Know when (and how) to use C++ in another course.

C++ - What you should have learned:

- What of C you can use in C++ and what C++ has to offer more (or in a different way) ...
- especially, the Object Oriented Programming Paradigm(!): Encapsulation, Polymorphism, Inheritance.
- Declaring classes and instantiating objects, accessing members, ...
- Understanding of 'conceptual programming', i.e. hiding of functions, declaring of static, const, virtual ...
- Understand C++ code if you see it.
- Know when (and how) to use C++ in another course.

In general:

- have an understanding for basic concepts in programming.
- be skilled enough using MATLAB, so it does not pose a problem in other courses.
- solve problems and implement algorithms in C and C++.
- be able to read and understand existing code written in C or C++.
- know the importance of writing code which others can understand, change, correct and build upon.

In general:

- have an understanding for basic concepts in programming.
- be skilled enough using MATLAB, so it does not pose a problem in other courses.
- solve problems and implement algorithms in C and C++.
- be able to read and understand existing code written in C or C++.
- know the importance of writing code which others can understand, change, correct and build upon.

In general:

- have an understanding for basic concepts in programming.
- be skilled enough using MATLAB, so it does not pose a problem in other courses.
- solve problems and implement algorithms in C and C++.
- be able to read and understand existing code written in C or C++.
- know the importance of writing code which others can understand, change, correct and build upon.

In general:

- have an understanding for basic concepts in programming.
- be skilled enough using MATLAB, so it does not pose a problem in other courses.
- solve problems and implement algorithms in C and C++.
- be able to read and understand existing code written in C or C++.
- know the importance of writing code which others can understand, change, correct and build upon.

In general:

- have an understanding for basic concepts in programming.
- be skilled enough using MATLAB, so it does not pose a problem in other courses.
- solve problems and implement algorithms in C and C++.
- be able to read and understand existing code written in C or C++.
- know the importance of writing code which others can understand, change, correct and build upon.

Summary

We have learnt a tool but we have not done much Computer Science yet

- Algorithms: *Sorting*, *Mapping*, ...
- Data structures: *Trees*, *Graphs*, ...
- Complexity
- Discrete Math
- ...

How to continue?

- The aim of this course was to get you started
- Hundreds of References and Books - to learn more and have a quick lookup for more specific things you need.
- Some more concentrated programming courses at KTH:
  - **DD2385** Programutvecklingsteknik
  - **DD2387** Programsystemkonstruktion med C++
  - **DD2390** Internet programming
  - **DD2257** Visualization
- Experience(!) - your own project.

Still to do:

- Your Evaluation
  - Finish C++-project submission
  - Final submission deadline **3** November
  - The course is only pass or fail
- Our Evaluation
  - Will be available through BILDA after the C++ project
  - For collecting feedback and opinions about the course.

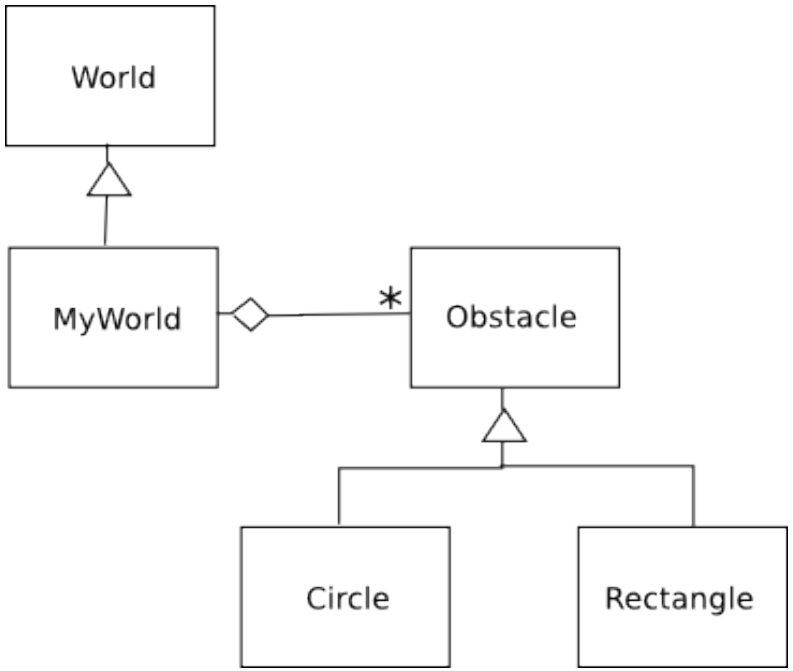
Still to do:

- Your Evaluation
  - Finish C++-project submission
  - Final submission deadline **3** November
  - The course is only pass or fail
- Our Evaluation
  - Will be available through BILDA after the C++ project
  - For collecting feedback and opinions about the course.

C++-Project

1. Make the provided code compile
2. Understand and identify the “flow” of the program
3. What extensions do I need to do?
4. Understand the `solvePlanningProblem.cc`
5. Understand the `SingleCircleWorld`
6. Whats the best structure for the functionality?

C++-Project: Suggestion



Next Time

- C++ Help Session, Tuesday L21 10-12
- Todo: Start with C++ project
- C++ project deadline Sunday 16/10