02393 Programming in C++
Module 2: C++ language features

Sebastian Mödersheim

September 5, 2016

Lecture Plan

#	Date	Topic
1	29.8.	Introduction
2	5.9.	Basic C++
3	12.9.	Data Types, Pointers
4	19.9.	
5	26.9.	Libraries and Interfaces; Containers
6	3.10.	Classes and Objects I
7	10.10.	Classes and Objects II
		Efterårsferie
8	24.10.	Classes and Objects III
9	31.10.	Recursive Programming
10	7.11.	Lists
11	14.11.	Trees
12	21.11.	Novel C++ features
13	28.11.	Summary
	5.12.	Exam

Disclaimer

General note on live programming:
On these lecture slides, we will not spell out all points covered and discussed in live programming sessions!

- We give the key words of the covered concepts
- We put the final version of the developed program on campusnet
- There are detailed explanations in many C++ books

Especially if you miss a live programming session, please make sure that you understand the material in detail, and ask questions to the TAs or in the next lecture!

Functions

Live programming session today will cover some of:

- Local variables, parameters
- Several functions
- Function prototypes
- Recursion

Functions

An Abstract View

- A bit like in mathematics:
  - ★ give an argument/several arguments
  - ★ get a result
- Differences—in C++ it is actually a procedure
  - ★ it can have side effects like printing on the screen
  - ★ it can depend on/change global variables
  - ★ thus: two calls with same arguments may produce different results
  - ★ functions may have no result at all:
    if return type is `void`
  - ★ Later: call by reference
- Scope: arguments and local variables are declared only for the body of the procedure

Bottom line: a good tool to break down a big problem into smaller ones.

Functions

A Result From Cognitive Psychology

- Comparing how experienced programmers and novices approach complex programming tasks.

Functions

A Result From Cognitive Psychology

- Comparing how experienced programmers and novices approach complex programming tasks.
- Break down problems in small ones (tree!)

Functions

A Result From Cognitive Psychology

- Comparing how experienced programmers and novices approach complex programming tasks.
- Break down problems in small ones (tree!)
- Novice approach:
  - ★ Begin with a sub-problem you can solve directly.
  - ★ Then try to iteratively extend the program to cover all requirements.

Functions

A Result From Cognitive Psychology

- Comparing how experienced programmers and novices approach complex programming tasks.
- Break down problems in small ones (tree!)
- Novice approach:
  - ★ Begin with a sub-problem you can solve directly.
  - ★ Then try to iteratively extend the program to cover all requirements.
- Expert approach:
  - ★ Define top-level problems before thinking about their solution.
  - ★ Reasoning style: Suppose I can solve these sub-problems, would that solve my main problem?
  - ★ Only after that has been cleared, go further into details.

Functions

A Result From Cognitive Psychology

- Comparing how experienced programmers and novices approach complex programming tasks.
- Break down problems in small ones (tree!)
- Novice approach:
  - ★ Begin with a sub-problem you can solve directly.
  - ★ Then try to iteratively extend the program to cover all requirements.
- Expert approach:
  - ★ Define top-level problems before thinking about their solution.
  - ★ Reasoning style: Suppose I can solve these sub-problems, would that solve my main problem?
  - ★ Only after that has been cleared, go further into details.
- Novices and experts can reach the same *beautiful* programs. However the novices waste a lot of time re-writing and overhauling their program again and again.

Functions

A Result From Cognitive Psychology

- Comparing how experienced programmers and novices approach complex programming tasks.
- Break down problems in small ones (tree!)
- Novice approach:
  - Begin with a sub-problem you can solve directly.
  - Then try to iteratively extend the program to cover all requirements.
- Expert approach:
  - Define top-level problems before thinking about their solution.
  - Reasoning style: Suppose I can solve these sub-problems, would that solve my main problem?
  - Only after that has been cleared, go further into details.
- Novices and experts can reach the same *beautiful* programs. However the novices waste a lot of time re-writing and overhauling their program again and again.
- A big project in expert mode may result in a long planning process without any executable program code. That makes many (managers) nervous!

Anderson, J.R.: *Cognitive Psychology and its Implications*, 7th edition, 2009.

Live Programming

Course repository for live programming

This course has a repository for live programming. Use svn checkout svn://repos.gbar.dtu.dk/samo/cpp2016/
with username `student` and password `yvyebbnnq532ej3b`

- Implementating several functions
- Using prototypes
- Using recursion
- Bounds of the data types

Live Programming

Course repository for live programming

This course has a repository for live programming. Use svn checkout svn://repos.gbar.dtu.dk/samo/cpp2016/
with username `student` and password `yvyebbnnq532ej3b`

- Implementating several functions
- Using prototypes
- Using recursion
- Bounds of the data types
- Exponentiation by squaring

Live Programming

Course repository for live programming

This course has a repository for live programming. Use svn checkout svn://repos.gbar.dtu.dk/samo/cpp2016/
with username `student` and password `yvyebbnnq532ej3b`

- Implementating several functions
- Using prototypes
- Using recursion
- Bounds of the data types
- Exponentiation by squaring

Other examples:

- $x - y + z = x + z - y$
- $(x + y)/2 = x/2 + y/2$

These equations may not always hold when working with C++ data types.

Live Programming

Course repository for live programming

This course has a repository for live programming. Use svn checkout svn://repos.gbar.dtu.dk/samo/cpp2016/
with username `student` and password `yvyebbnnq532ej3b`

- Implementating several functions
- Using prototypes
- Using recursion
- Bounds of the data types
- Exponentiation by squaring

Other examples:

- $x - y + z = x + z - y$
- $(x + y)/2 = x/2 + y/2$

These equations may not always hold when working with C++ data types.
Bottom line: Be aware of the limits of the used data types!

Functions

A Technical View

On the machine level, nested procedure calls are implemented using a stack:

- Basically, the stack keeps track of where to continue after a procedure has finished, i.e., to which computation to return to.
- For each function call, the system puts a new frame on top of the stack.
- For each finished function, the system removes the top frame from the stack (restoring the data from the previous stack frame and “jumping” to the right point in the code).
- A frame consists of the return address and the local variables of the current procedure.
- The size of the stack is limited, so
  - ★ excessive nested procedure calls
  - ★ excessive amount of local variables

  can lead to a stack overflow.
- Arguments and results are copied (when using call by value as we did so far): the local variables of the calling procedure are not affected!

Exercises and CodeJudge

- There is an exercise sheet on campusnet filesharing
- Hand-in via CodeJudge until next Monday before the lecture.