INF200 Advanced Programming

Hans Ekkehard Plesser

Department of Mathematical Sciences and Technology
Norwegian University of Life Sciences

22 January 2013

Textbook

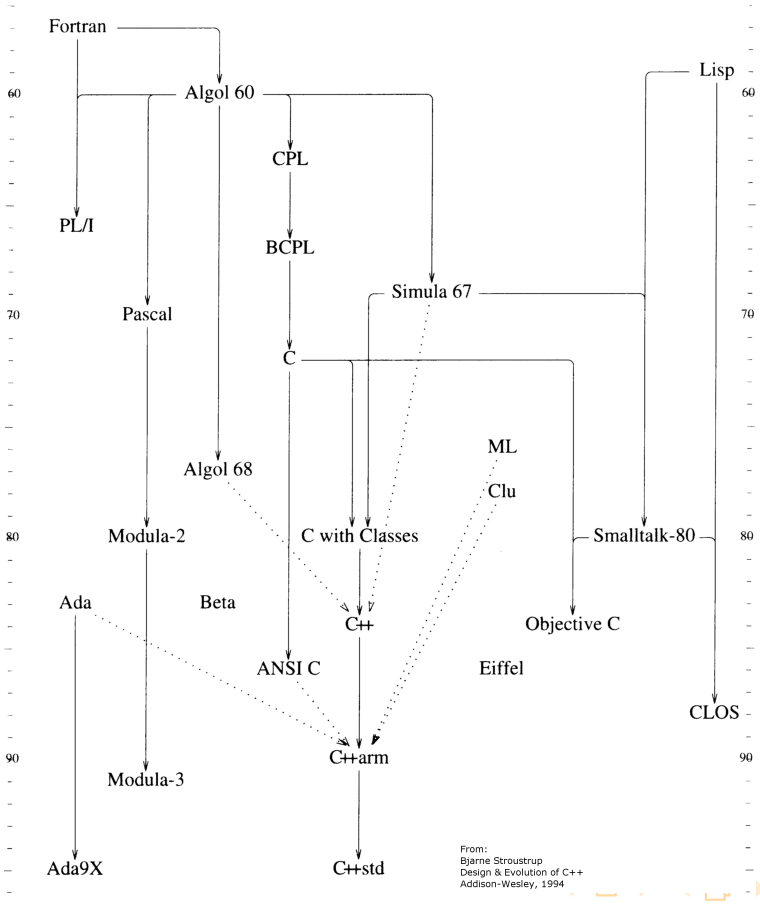
Andrew Koenig and Barbara E. Moo,
Accelerated C++: Practical Programming by Example.
Addison-Wesley, 2000. ISBN 0-201-70353-X.

The C++ Programming Language

The C++ Programming Language

Properties of C++

History Developed by B. Stroustrup ca 1980–90
Based on C and Simula
Standardized in 1998 (ISO 14882)
New C++ standard in 2011



Properties of C++

History Developed by B. Stroustrup ca 1980–90
Based on C and Simula
Standardized in 1998 (ISO 14882)
New C++ standard in 2011

Use Among most used languages
“Backend” rather than “front end”

Advantages Highly flexible
Support procedural, object oriented, and generic programming
Very efficient compilers
Stort brukermiljø
STL gives programmer large toolchest

Disadvantages Large language: impossible to learn all
“Everything goes”, but it is you responsibility. . .
No graphics support built in

Properties of C++

History Developed by B. Stroustrup ca 1980–90
Based on C and Simula
Standardized in 1998 (ISO 14882)
New C++ standard in 2011

Use Among most used languages
“Backend” rather than “front end”

Advantages Highly flexible
Support procedural, object oriented, and generic programming
Very efficient compilers
Stort brukermiljø
STL gives programmer large toolchest

Disadvantages Large language: impossible to learn all
“Everything goes”, but it is you responsibility. . .
No graphics support built in

Properties of C++

History Developed by B. Stroustrup ca 1980–90
Based on C and Simula
Standardized in 1998 (ISO 14882)
New C++ standard in 2011

Use Among most used languages
“Backend” rather than “front end”

Advantages Highly flexible
Support procedural, object oriented, and generic programming
Very efficient compilers
Stort brukermiljø
STL gives programmer large toolchest

Disadvantages Large language: impossible to learn all
“Everything goes”, but it is you responsibility. . .
No graphics support built in

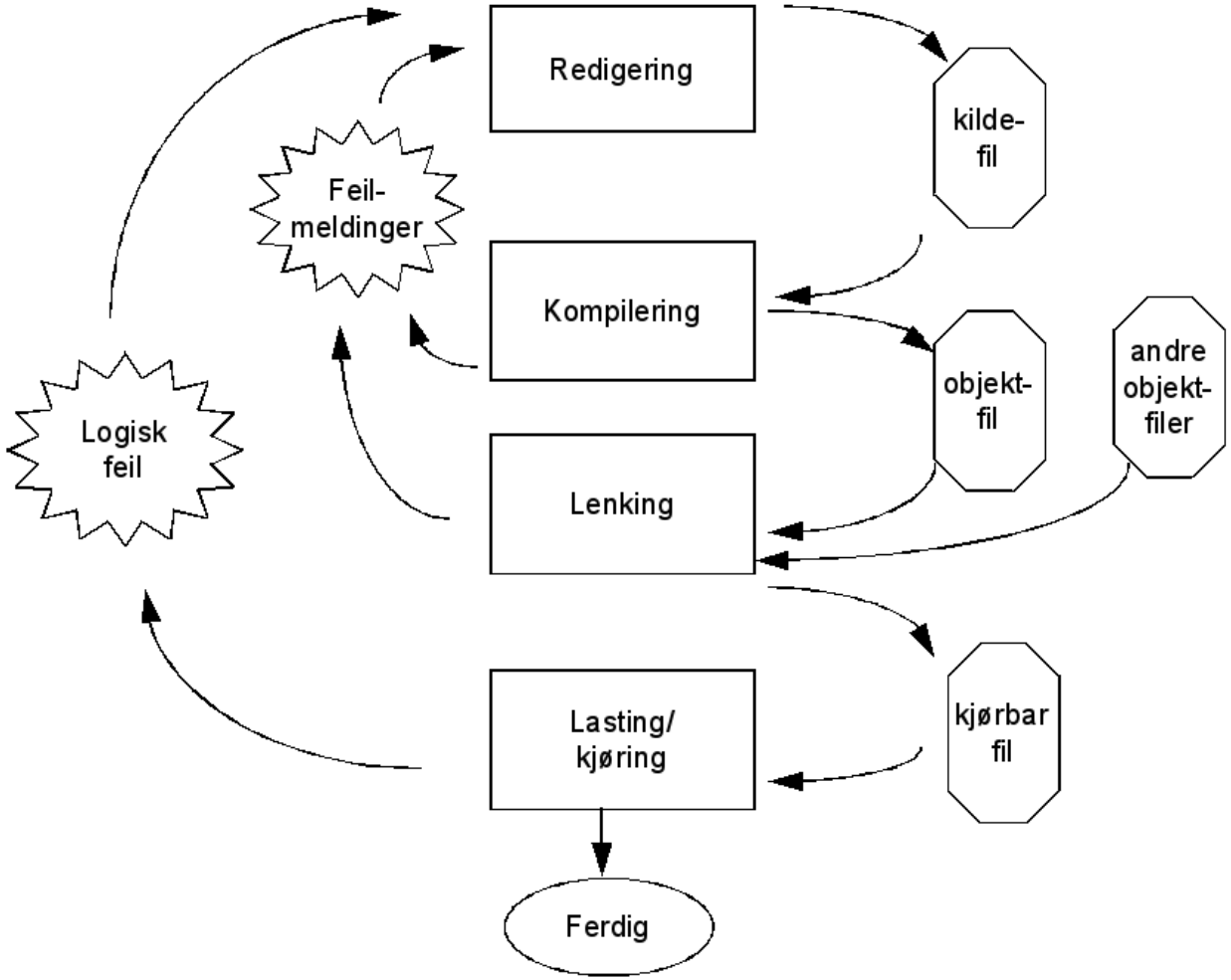
Our first program

```
/*
   A small C++ program.
   Author: Hans Ekkehard Plesser
*/

#include <iostream>

int main()
{
  // print one line
  std::cout << "Hello, world!" << std::endl;
  return 0;
}
```

From code to executable



from Lervik/Ljosland

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

What's in the program?

Comment	`/* ... */  // ...`
Header files	`#include <iostream>`

Main program	`int main ()`
	`{`
	`}`

String literals	`"This is a string literal."`
Characters	`'a','b','$',...`
Special chars	`\n, \t, \b, \", \', \\`
Output	`std::cout << ... << std::endl;`
Namespace	`std`

Statements and Expressions

Statements and Expressions

Statements and Expressions

- programs are composed of statements
- statements end with a `;`
- a block combines several statements
- blocks are delimited with `{` and `}`
- within functions, blocks can be used wherever statements can be used

Input and output

Input and output

Input and output in C++

- C++ itself does not support I/O
- I/O support is provided by the standard library (STL)
- I/O has to be included with `#include`

`iostream`	input/output from keyboard/screen
`iomanip`	formatting of output
`fstream`	access to files

- all I/O components are in namespace `std`

I/O streams & operators

streams	in C++, I/O is via *streams* connected to devices/files
`std::istream`	data type for input stremas
`std::ostream`	data type for output streams
`std::cin`	standard input stream (keyboard)
`std::cout`	standard output stream (screen)
`std::cerr`	output stream for error messages (screen)
`std::endl`	manipulator: new line
`<<`	output operator (formatted output)
`>>`	input operator

```
// ask for name, greet user
#include <iostream>
#include <string>

int main()
{
    // ask for user's name
    std::cout << "Please enter your first name: ";

    // read the name
    std::string name;   // define name
    std::cin >> name;   // read it into name

    // print greeting
    std::cout << "Hello, " << navn << "!" << std::endl;

    return 0;
}
```

About variables

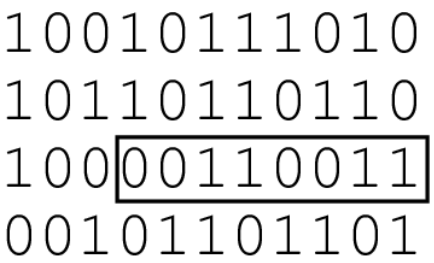
About variables

- used to store data in memory
  ↪ variable: an object with a name
  ↪ object: a part of memory with data type
  ↪ data type: a rule for interpreting data in a part of the memory

```
10010111010
10110110110
10000110011
00101101101
```

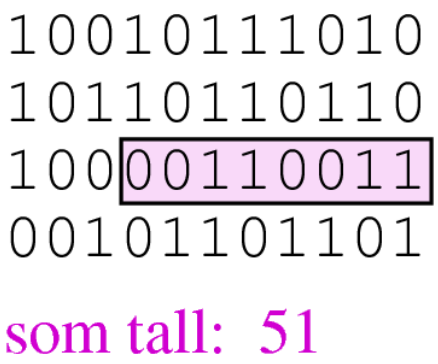
About variables

- used to store data in memory
  ↪ variable: an object with a name
  ↪ object: a part of memory with data type
  ↪ data type: a rule for interpreting data in a part of the memory



About variables

- used to store data in memory
  ↪ variable: an object with a name
  ↪ object: a part of memory with data type
  ↪ data type: a rule for interpreting data in a part of the memory



About variables

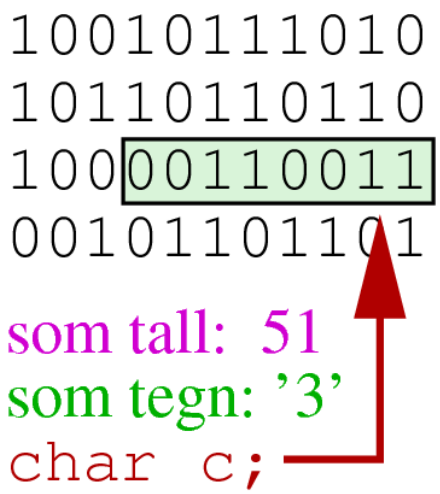
- used to store data in memory
  ↪ variable: an object with a name
  ↪ object: a part of memory with data type
  ↪ data type: a rule for interpreting data in a part of the memory

```
10010111010
10110110110
10000110011
00101101101
```

som tall: 51
som tegn: '3'

About variables

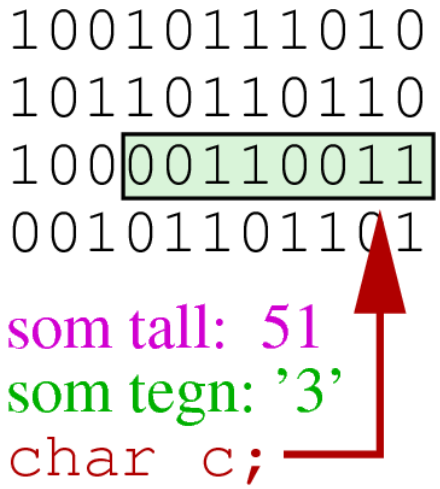
- used to store data in memory
  ↪ variable: an object with a name
  ↪ object: a part of memory with data type
  ↪ data type: a rule for interpreting data in a part of the memory



About variables

- used to store data in memory
  ↪ variable: an object with a name
  ↪ object: a part of memory with data type
  ↪ data type: a rule for interpreting data in a part of the memory
- variables must be defined

```
char c;
int n;
double x;
std::string name;
```



Further illustraion by program examples.