1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace `std`

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements

MCS 360 Lecture 1
Introduction to Data Structures
Jan Verschelde, 13 January 2020
Introduction to Data Structures

1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace std

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements
MCS 360 - Introduction to Data Structures

Pointers and dynamic memory allocation in C/C++, recursion, stacks, queues, heaps, binary and multiway trees, graphs, hash tables. Sorting and searching algorithms.

Prerequisite(s): Grade of C or better in MCS 260 and Grade of C or better in MCS 275.

MCS 275 is still required.
The design of the course follows

Reading the textbook is recommended. However, programming is a skill, acquired mainly by practice, practice, and lots of practice...

A natural sequel to MCS 360 is MCS 401/CS 401: computer algorithms.
Organization of the Material
along the text book

Five parts:

1. introduction to C++ (chapters P, 1, 2, 3)
2. vector, stack, list, queue and deque (chap 4, 5, 6)

*first midterm on chapters P, 1, 2, 3, 4, 5, 6*

3. recursion, trees, maps, hash tables (chap 7, 8, 9)
4. sorting algorithms, balancing trees (chap 10, 11)

*second midterm on chapters 7, 8, 9, 10, 11*

5. after the second midterm, we cover chapter 12 on graphs.
Introduction to Data Structures

1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace std

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements
Purpose of the Course

Goals of MCS 360 are threefold:
1. solve problems with algorithms using *right* data structures (programs = algorithms + data structures)
2. gain basic working knowledge of C++
3. application of the Standard Template Library (STL)

About C++
- widespread use with good performance
- no framework language as Java or Python

Some important points:
- emphasis on five computer projects,
- active participation to the lab sessions.
Introduction to Data Structures

1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace `std`

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements
The GNU compiler collection includes front ends for C, C++, ...

\texttt{g++} = GNU C++, a freely redistributable C++ compiler

By default installed on Unix, Linux, Mac OS X.

On windows: www.mingw.org or www.cygwin.com

Although object-oriented programming supports programming in the large, our programs will typically remain small.

C++ is a compiled language

The compiler translates C++ code in hello_world.cpp into code the computer can execute.

At the command prompt $ we type

$ g++ -o hello_world hello_world.cpp

The output of g++ is renamed into hello_world, with the extension .exe on Windows computers.

To run the program, type at the command prompt $

$ hello_world
Hello World!
$

Any editor can be used to write .cpp files.
our first program

Our first program writes Hello World! to screen.

A C++ program typically consists of

1. preprocessor directives
2. function definitions
3. type declaration of variables
4. control statements

Recommended way of working:

1. write basic version of code
2. compile and test, if done then stop
3. extend adding extra statements
4. go to step 2
About the Course
- content
- expectations

our first C++ program
- using g++
- input and output streams
- the namespace std

Greatest Common Divisor
- Euclid’s algorithm
- the while and do-while statements
code for our first program

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

1. preprocessor directives start with 
2. including `iostream` is needed for the insertion operator `<<`.  
3. the program is the definition of a function `main`  
4. one statement sends `Hello World!` to screen, followed by the end of line (endl) symbol  
5. the function returns zero if all went well
Comments and Blocks

The compiler ignores lines starting with //, e.g.:

```
// L-1 MCS 360 Mon 13 Jan 2020 : hello_world.cpp
```

Multiple lines of documentation are between /* and */.

Code between curly braces defines a block.

```
int main()
{
    // omitted code
    return 0;
}
```

Every statement is terminated by semicolon ;
indentation is optional, but strongly recommended.
Our first interactive program

1. prompts the user to enter a name,
2. stores the entered name in a string,
3. writes a personalized greeting.

If we save the program in `hello_there`
we run it at the prompt $ as

$$
$ hello_there
Who is there ? me
Hello me!
$
Introduction to Data Structures

1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace std

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements
Instead of

```cpp
std::cout << "Hello World!" << std::endl;
```

we can write

```cpp
cout << "Hello World!" << endl;
```

if we insert

```cpp
using namespace std;
```

after the preprocessor directives.

A *namespace* is a collection of names or identifiers defined together (like a Python module). The C++ standard library is defined in namespace `std`. 
```cpp
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string name;

    cout << "Who is there ? "; // insertion operator
    cin >> name;       // >> is the extraction operator
    cout << "Hello " << name << "!" << endl;

    return 0;
}
```
Exercise 1:

One kilogram (kg) is 2.20462 pounds (lb).

Write a C++ program which prompts the user for a weight in kilograms. The program computes the weight measured in pounds and writes the corresponding pounds to the standard output.

A session with the program could go as follows:

Give a weight in kg : 50
The weight 50 in pounds is 110.231.

Note: the type for a floating-point number is double.
Introduction to Data Structures

1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace std

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements
Euclid’s Algorithm

Input: a, b; \( a \geq b > 0 \).
Output: gcd of a and b.

\( r := \text{remainder of } a/b \)

\( r = 0? \)

Yes
print b

No
\( a := b; \ b := r \)
computing a GCD

$\text{gcd\_dowhile1}$

Welcome to our Greatest Common Divisor

give a positive integer $x : 468$
give a positive integer $y : 864$
computing the gcd of 468 and 864 ...

$x = 468 \quad y = 864 \quad r = 468$
$x = 864 \quad y = 468 \quad r = 396$
$x = 468 \quad y = 396 \quad r = 72$
$x = 396 \quad y = 72 \quad r = 36$
$x = 72 \quad y = 36 \quad r = 0$

$\text{gcd}(468,864) = 36$

$\$

- confirm input before and after computations
- intermediate output to monitor progress
Introduction to Data Structures

1. About the Course
   - content
   - expectations

2. our first C++ program
   - using g++
   - input and output streams
   - the namespace std

3. Greatest Common Divisor
   - Euclid’s algorithm
   - the while and do-while statements
Either first test the condition in `while` **before** executing the statements:

```c
< initialization >
while( < condition > )
{
    < statements >
}
```

Or test the condition in `while` **after** executing the statements:

```c
< initialization >
do
{
    < statements >
} while( < condition > );
```
#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    int x,y;

cout << "Welcome to our Greatest Common Divisor\n";
cout << " give a positive integer x : ";
cin >> x;
cout << " give a positive integer y : ";
cin >> y;
cout << "computing the gcd of "
    << x << " and " << y << " ..." << endl;

ostringstream s; // write to a string
s << "gcd(" << x << "," << y << ") = ";

}
computing the GCD

```cpp
int r;

do {
    r = x % y; // remainder calculation
    cout << " x = " << x
         << " y = " << y
         << " r = " << r << endl;
    x = y;
    y = r;
} while (r != 0); // stop test

cout << s.str() << x << endl;

return 0;
}```
computing the binary expansion of a number

Exercise 2:

Write a C++ program which prompts the user for a positive integer number.
The program writes the input number and prints the bits in the binary decomposition of the number, in reverse order, printing the least significant bit first.

A session with the program could go as follows:

Give a number : 360
The bits in 360 : 0 0 0 1 0 1 1 0 1.
In this lecture we covered first three sections of Chapter P.

On Tuesday or Thursday go to the computer lab,
- make sure your netid is working,
- there will be a quiz at the end.

Additional exercises:

3. Install `g++` on your laptop or home computer.

4. Do `g++ -E hello_world.cpp` (our first C++ program). What happens? What is this option `-E`?

5. Extend `gcd_dowhile1.cpp` with a check that the greatest divisor reported after the loop really divides the two given numbers.

6. Instead of `do-while`, use a `while-do` statement to encode Euclid’s algorithm.