

# Welcome to MCS 360

## 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

the while and do-while statements

- 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 Vershelde, 23 August 2010

# Introduction to Data Structures

## 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

### 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

# Catalog Description

more or less up to date

## 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.

Changes:

- 1 **MCS 275 no longer required, because of Python**
- 2 *Any Introduction to Computer Science (MCS 260) will do just as well as prerequisite.*

# Catalog Description

more or less up to date

## 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.

Changes:

- 1 **MCS 275 no longer required, because of Python**
- 2 Any *Introduction to Computer Science* (MCS 260) will do just as well as prerequisite.

# Content of the Course

Text Book

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

The design of the course follows

Elliot B. Koffman and Paul A.T. Wolfgang:  
*Objects, Abstraction, Data Structures, and Design  
using C++*, John Wiley and Sons, 2006.

A natural sequel to MCS 360  
is MCS 401/CS 401: computer algorithms.

At the end of the course, we follow chapter 4 of  
T. H. Cormen, C. E. Leiserson, R. L. Rivest, C, Stein:  
*Introduction to Algorithms* (2nd edition at googlebooks).

# Content of the Course

Text Book

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

The design of the course follows

Elliot B. Koffman and Paul A.T. Wolfgang:  
*Objects, Abstraction, Data Structures, and Design  
using C++*, John Wiley and Sons, 2006.

A natural sequel to MCS 360  
is MCS 401/CS 401: computer algorithms.

At the end of the course, we follow chapter 4 of  
T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein:  
Introduction to Algorithms (2nd edition at googlebooks).

# Content of the Course

Text Book

The design of the course follows

Elliot B. Koffman and Paul A.T. Wolfgang:

*Objects, Abstraction, Data Structures, and Design using C++*, John Wiley and Sons, 2006.

A natural sequel to MCS 360

is MCS 401/CS 401: computer algorithms.

At the end of the course, we follow chapter 4 of

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C, Stein:

Introduction to Algorithms (2nd edition at googlebooks).

# 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 methods to solve recurrences (chap 4 of 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 methods to solve recurrences (chap 4 of 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 methods to solve recurrences (chap 4 of algorithms)

# Introduction to Data Structures

## 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

### 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 basics of discrete math to analyze algorithms

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.

# 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 basics of discrete math to analyze algorithms

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.

# Purpose of the Course

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

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 basics of discrete math to analyze algorithms

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

## 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

### 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

# using g++

free available open source

The GNU compiler collection includes front ends for C, C++,

...

g++ = GNU C++, a freely redistributable C++ compiler

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

On windows: [www.mingw.org](http://www.mingw.org) or [www.cygwin.com](http://www.cygwin.com)

Although object-oriented programming supports programming in the large, programs for this class will typically be small, and fit on one page.

# using g++

free available open source

The GNU compiler collection includes front ends for C, C++,

...

`g++` = GNU C++, a freely redistributable C++ compiler

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

On windows: [www.mingw.org](http://www.mingw.org) or [www.cygwin.com](http://www.cygwin.com)

Although object-oriented programming supports programming in the large, programs for this class will typically be small, and fit on one page.

# C++ is a compiled language

## 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

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

At the command prompt `$` we type

```
$ g++ -o /tmp/hello_world hello_world.cpp
```

The output of `g++` is at `/tmp/hello_world`.

To run the program, type at the command prompt `$`

```
$ /tmp/hello_world  
Hello World!  
$
```

Any editor can be used to write `.cpp` files.

# C++ is a compiled language

## 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

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

At the command prompt `$` we type

```
$ g++ -o /tmp/hello_world hello_world.cpp
```

The output of `g++` is at `/tmp/hello_world`.

To run the program, type at the command prompt `$`

```
$ /tmp/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

# 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

# Introduction to Data Structures

## 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

### 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

# code for our first program

## 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

```
#include <iostream>

int main()
{
    std::cout << "Hello World!" << std::endl;

    return 0;
}
```

- 1 preprocessor directives start with #
- 2 the program is the definition of a function `main`
- 3 one statement sends `Hello World!` to screen, followed by the end of line (`endl`) symbol
- 4 the function returns zero if all went well

# code for our first program

## 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

```
#include <iostream>

int main()
{
    std::cout << "Hello World!" << std::endl;

    return 0;
}
```

- 1 preprocessor directives start with #
- 2 the program is the definition of a function `main`
- 3 one statement sends `Hello World!` to screen, followed by the end of line (`endl`) symbol
- 4 the function returns zero if all went well

# code for our first program

## 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

```
#include <iostream>

int main()
{
    std::cout << "Hello World!" << std::endl;

    return 0;
}
```

- 1 preprocessor directives start with #
- 2 the program is the definition of a function `main`
- 3 one statement sends `Hello World!` to screen, followed by the end of line (`endl`) symbol
- 4 the function returns zero if all went well

# Comments and Blocks

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

```
// L-1 MCS 360 Mon 23 Aug 2010 : 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.

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

# Comments and Blocks

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

```
// L-1 MCS 360 Mon 23 Aug 2010 : 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

## 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

## 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 `/tmp/hello_there`  
we run it at the prompt `$` as

```
$ /tmp/hello_there  
Who is there ? me  
Hello me!  
$
```

# our first interactive program

## 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

## 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 `/tmp/hello_there`  
we run it at the prompt `$` as

```
$ /tmp/hello_there  
Who is there ? me  
Hello me!  
$
```

# Introduction to Data Structures

## 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

### 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 namespace `std`

## Instead of

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

## we can write

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

## if we insert

```
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`.

# the namespace `std`

Instead of

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

we can write

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

if we insert

```
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`.

# the namespace `std`

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

Instead of

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

we can write

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

if we insert

```
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`.

## code for the entire program

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string name;

    cout << "Who is there ? ";
    cin >> name;
    cout << "Hello " << name << "!" << endl;

    return 0;
}
```

## code for the entire program

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string name;

    cout << "Who is there ? ";
    cin >> name;
    cout << "Hello " << name << "!" << endl;

    return 0;
}
```

# Introduction to Data Structures

## 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

### 1 About the Course

content  
expectations

### 2 our first C++ program

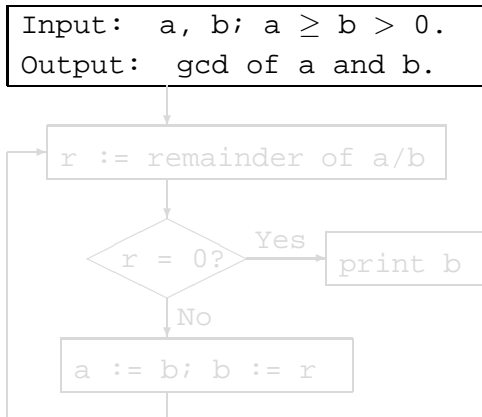
using g++  
input and output streams  
the namespace `std`

### 3 Greatest Common Divisor

**Euclid's algorithm**  
the while and do-while statements

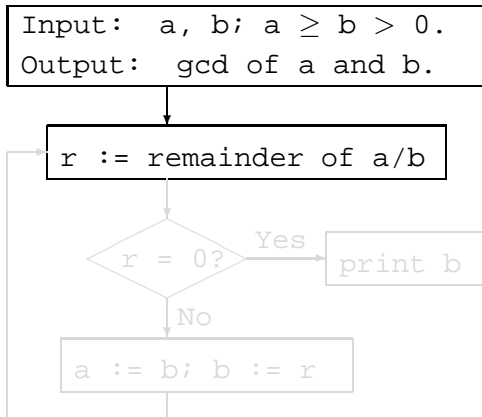
# Euclid's Algorithm

flowchart



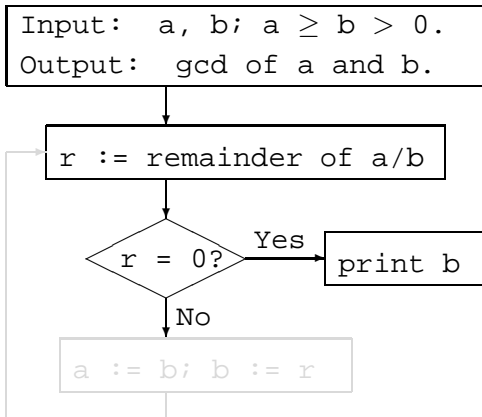
# Euclid's Algorithm

flowchart



# Euclid's Algorithm

flowchart



About the  
Course

content  
expectations

our first C++  
program

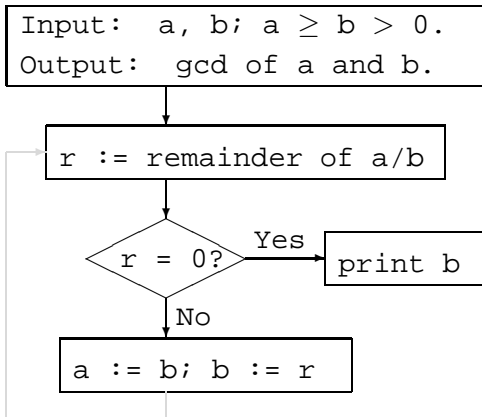
using g++  
input and output  
streams  
the namespace `std`

Greatest  
Common  
Divisor

Euclid's algorithm  
the while and  
do-while statements

# Euclid's Algorithm

flowchart



About the  
Course

content  
expectations

our first C++  
program

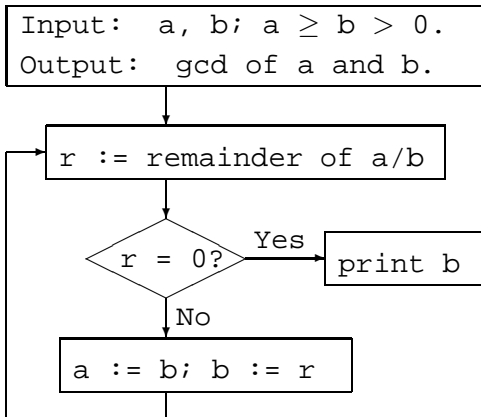
using g++  
input and output  
streams  
the namespace `std`

Greatest  
Common  
Divisor

Euclid's algorithm  
the while and  
do-while statements

# Euclid's Algorithm

flowchart



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

## computing a GCD

## 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

```
$ /tmp/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 y = 864 r = 468
```

```
x = 864 y = 468 r = 396
```

```
x = 468 y = 396 r = 72
```

```
x = 396 y = 72 r = 36
```

```
x = 72 y = 36 r = 0
```

```
gcd(468,864) = 36
```

```
$
```

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

## computing a GCD

## 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

```
$ /tmp/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  y = 864  r = 468
  x = 864  y = 468  r = 396
  x = 468  y = 396  r = 72
  x = 396  y = 72   r = 36
  x = 72   y = 36   r = 0
gcd(468,864) = 36
$
```

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

## computing a GCD

```
$ /tmp/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  y = 864  r = 468
  x = 864  y = 468  r = 396
  x = 468  y = 396  r = 72
  x = 396  y = 72   r = 36
  x = 72   y = 36   r = 0
gcd(468,864) = 36
$
```

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

# Introduction to Data Structures

## 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

### 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

# while and do-while

Either first test the condition in `while`  
**before** executing the statements:

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

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

```
< initialization >
do
{
    < statements >
} while( < condition > );
```

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

# while and do-while

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

Either first test the condition in `while`  
***before*** executing the statements:

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

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

```
< initialization >
do
{
    < statements >
} while( < condition > );
```

# while and do-while

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

Either first test the condition in `while`  
**before** executing the statements:

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

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

```
< initialization >  
do  
{  
    < statements >  
} while( < condition > );
```

# while and do-while

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

Either first test the condition in `while`  
**before** executing the statements:

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

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

```
< initialization >  
do  
{  
    < statements >  
} while( < condition > );
```

## reading numbers

```
#include <iostream>
#include <sstream>
using namespace std;
```

```
int main()
{
```

```
    int x,y;
```

```
    cout << "Welcome to our Greatest Common Divisor\
```

```
    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 << ") = ";
```

## reading numbers

```

#include <iostream>
#include <sstream>
using namespace std;

int main()
{
    int x,y;

    cout << "Welcome to our Greatest Common Divisor\
    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 << ") = ";

```

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

## reading numbers

About the  
Coursecontent  
expectationsour first C++  
programusing g++  
input and output  
streams  
the namespace `std`Greatest  
Common  
DivisorEuclid's algorithm  
the `while` and  
`do-while` statements

```
#include <iostream>
#include <sstream>
using namespace std;
```

```
int main()
{
```

```
    int x,y;
```

```
    cout << "Welcome to our Greatest Common Divisor\
```

```
    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

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](#)

```
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;
}
```

# Summary + Assignments

In this lecture we covered first three sections of Chapter P.

On Tuesday go to lab SEL 2263

- make sure your netid is working,
- there will be a quiz at the end.

## Assignments:

- 1 Install `g++` on your laptop or home computer.
- 2 Do `g++ -E hello_world.cpp` (our first C++ program). What happens? What is this option `-E`?
- 3 Extend `gcd_dowhile1.cpp` with a check that the greatest divisor reported after the loop really divides the two given numbers.
- 4 Instead of `do-while`, use a simple `while` statement to encode Euclid's algorithm.