Recitation 6

Overview

- Namespace
- Friend
- Using command line
- Git

The name conflict problem

- Imagine we want to define our own class **`vector`**

```
#include <vector>

template<class T>
class vector {
public:
    …
private:
    std::vector<T> m_data;
};
```

- STL vector lives in namespace std
- namespace has to be resolved

The name conflict problem

- For convenience, we want to resolve the namespace **`std`** generally

```
#include <vector>

using namespace std;

template<class T>
class vector {
public:
    …
private:
    vector<T> m_data;
};
```

- Name conflict!
- Compiler can't decide which vector to use
- Compiler won't allow the name **`vector`** for a class

The name conflict problem

- Other sources of name conflicts:
  - Variable name used more than once

```
#include <stdlib.h>

int lengthParameter;
int lengthParameter;

//…
```

Global namespace

The name conflict problem

- Other sources of name conflicts:
  - Function name/signature used more than once

```
#include <stdlib.h>

int computeLength();
int computeLength();

//…
```

Global namespace

Introducing your own namespace

- Example:

Custom namespace 1

Custom namespace 2

```
#include <iostream>

namespace VarSet1 {
    int lengthParameter;
    int computeLength();
}

namespace VarSet2 {
    int lengthParameter;
    int computeLength();
}

int main() {
    std::cout << VarSet1::lengthParameter << "\n";

}
```

Introducing your own namespace

- Namespaces for classes
  - A way to create logical grouping

```
namespace MyMathLibrary {
    template<class T>
    class Vector {
        //…
    };

    template<class T>
    class Matrix {
        //…
    };
}
```

Introducing your own namespace

- Namespaces for classes
  - A way to create logical grouping
  - Namespace additions can be in different files

File 1

```
namespace MyMathLibrary {
    template<class T>
    class Vector {
        //…
    };
}
```

File 2

```
namespace MyMathLibrary {
    template<class T>
    class Matrix {
        //…
    };
}
```

  - Custom namespaces can be generally included, too:
    `using namespace MyMathLibrary;`

Nested namespaces

- Namespaces can be nested
  - Creates a hierarchy of the functionality

```
namespace MyMathLibrary {
    namespace Linear {
        //…
    }
    namespace Nonlinear {
        namespace ClosedForm {
            //…
        }
        namespace Iterative {
            //…
        }
    }
}
```

  - Use: `using namespace MyMathLibrary::Nonlinear::Iterative::…;`

Friend

- Occasionally, we may want to allow a function that is not member of a given class to access its private fields/ methods

- Can specify that a given external function gets full access by
  - Placing the signature of the external function inside the class
  - Preceding this signature copy by the keyword

Function

```
class USCurrency {
   friend ostream& operator<<( ostream &o, const USCurrency &c);

public:
   USCurrency( const int d, const int c) : dollars(d), cents(c) {}

private:
   int dollars, cents;

};

ostream& operator<<( ostream &o, const USCurrency &c) {

   o << '$' << c.dollars << '.' << c.cents;
   return o;

}
```

```
int main(){

    UScorrency money(9,15);

    cout<<money;

}
```

Classes

Can do the same with classes

- Declaring other class as “friend” lets this class directly access private members of the other class

```
class A {
        friend class B;
        // More code …
};
```

Programming languages are advanced ways to control your computer with language.

The command line is the baby little brother of programming languages.Learning the command line teaches you to control the computer using language.

–someone

setup

macOS

Ask Siri...

Linux

I'm assuming that if you have Linux then you already know how to get at your terminal

Windows

- **Click Start.**
- **In "Search programs and files" type: powershell**
- **Hit Enter.**

Paths, Folders, Directories (pwd)

If You Get Lost

Make a Directory (mkdir)

Change Directory (cd)

List Directory (ls)

Remove Directory (rmdir)

Making Empty Files (touch, New-Item)

Copy a File (cp)

View a File (less, more)

Stream a File (cat)

Removing a File (rm)

Installation of git …

… under Ubuntu

- *sudo apt-get install git*

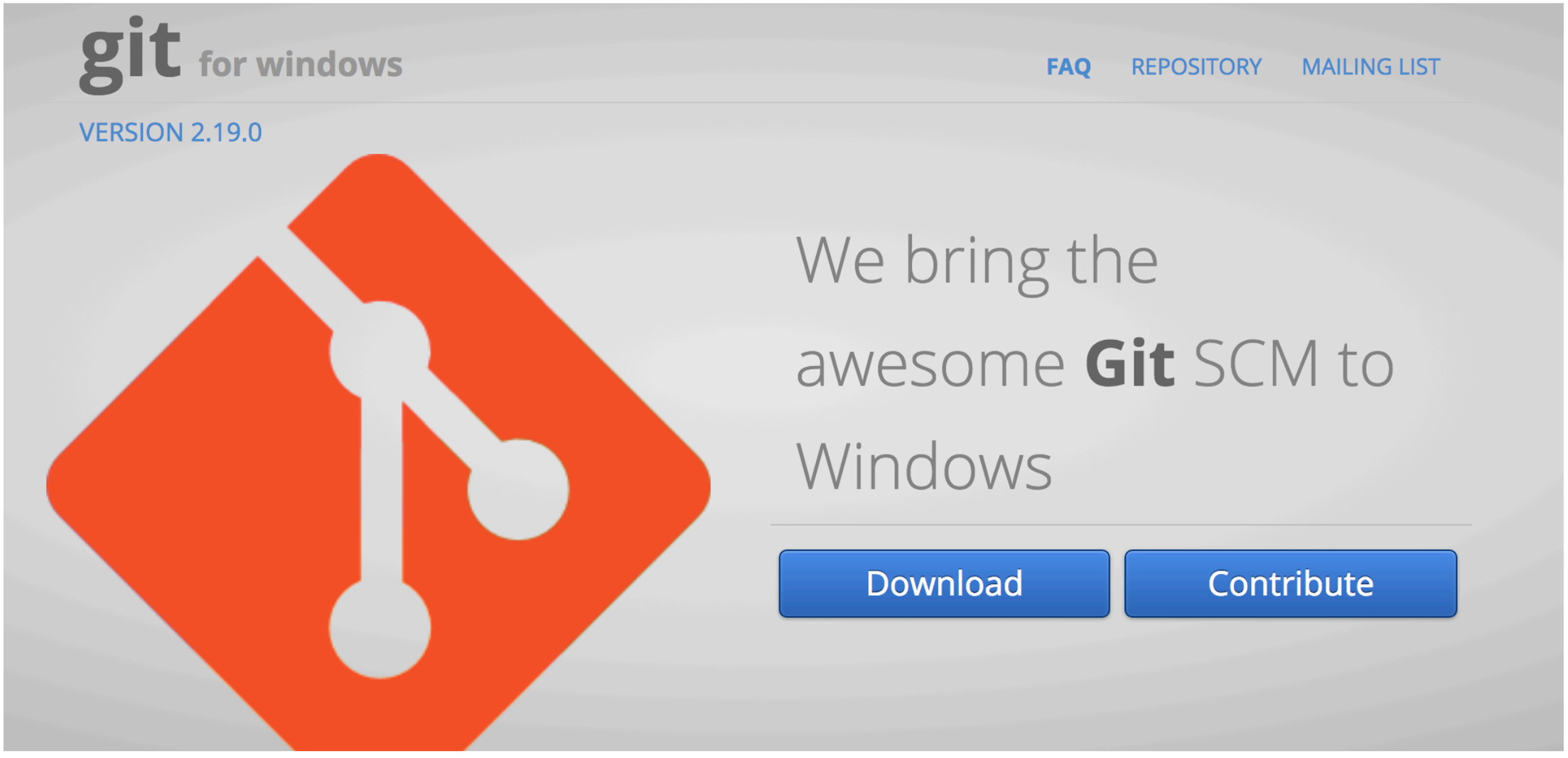
… under OSX

- Install brew (package manager)
  - *ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)" brew doctor*

- Install git
  - *brew install git*

… under Windows

- Download and install “git for windows”:
  - https://gitforwindows.org/



Basic git command

- clone
- pull
- push
- commit
- add

Using git

- Local changes can be ”saved” by a “commit”
  - *git commit <file-to-commit> –m “Some message”*
- New files have to be added to the repository
  - *git add <file-to-add>*
- Commits can be pushed to the cloud
  - *git push*
- Pull changes on the cloud (e.g. HW addition)
  - *git pull*
  - **Attention:** local changes have to be committed before pull