

Classes in C++

- Version 1: Dr. Ofir Pele
- Version 2: Dr. Miri Ben-Nissan
- Version 3: Dr. Erel Segal-Halevi

	C	C++	Java
Keyword	struct	class or struct	class
Filename	any (usually: <i>name.h</i>)	any (usually: <i>name.hpp</i> <i>name.cpp</i>)	<i>name.java</i>
Attributes	Yes	Yes	Yes
Methods	No	Yes	Yes
Access control	all public	public or private	public or private
Memory	stack	stack	heap
Operators	No	Yes	No

structs and classes

Where did **structs** go?

- In C++ **class**==**struct**, except that by default **struct** members are **public** and **class** members are **private**:

```
struct MyStruct
{
    int x;
};
class MyClass
{
    int x;
};
```

```
int main()
{
    MyStruct s;
    s.x = 1; // ok
    MyClass c;
    c.x = 1; // error
}
```

structs & classes *(folder 1)*

All of these are the same:

```
struct A
{
    int x;
};
```

```
struct A
{
    public:
    int x;
};
```

```
class A
{
    public:
    int x;
};
```

All of these are the same (and useless):

```
class A
{
    int x;
};
```

```
class A
{
    private:
    int x;
};
```

```
struct A
{
    private:
    int x;
};
```

Arrangement of Classes in Memory

- Version 1: Dr. Ofir Pele
- Version 2: Dr. Miri Ben-Nissan
- Version 3: Dr. Erel Segal-Halevi

C

```
struct Cplx {  
    double re, im;  
};
```

```
Cplx sumCplx(  
    Cplx a, Cplx b)  
{...}
```

C++

```
class Cplx {  
    double re, im;  
public:  
    Cplx sum  
        (Cplx b) {...}  
    Cplx  
        (double re,  
         double im) {...}  
};
```

Java

```
class Cplx {  
    private double  
        re, im;  
    public Cplx sum  
        (Cplx b) {...}  
    public Cplx  
        (double re,  
         double im) {...}  
};
```

C

```
int main() {  
    Cplx a;  
    a.re=5;  
    a.im=10;  
}
```

C++

```
int main() {  
    Cplx a(5,10);  
}
```

Java

```
void main(...) {  
    Cplx a =  
        new  
            Cplx(5,10);  
}
```

C, C++

Java

Stack:

a.re

a.im

b.re

b.im

c.re

c.im

Heap:

a

b

c

c.re

c.im

b.re

b.im

a.re

a.im

```
int main () { Cplx a, b, c; };
```


Two ways to implement a method *(folder 2)*

```
class Complex {  
    double re, im;  
public:  
    Complex () { re=0; im=0; } // inline constructor  
    Complex (double re, double im);    // “outline”  
  
    Complex sum (Complex b) { return  
Complex(a.re+b.re, a.im+b.im); } // inline method  
    Complex diff (Complex b);        // “outline”  
};
```

Implementing methods out-of-line

```
Complex::Complex (double re, double im) {
```

```
    this→re = re;
```

```
    this→im = im;
```

```
}
```

Scope operator

The address of the instance
for which the member
method was invoked.

```
Complex Complex::diff(Complex b) {
```

```
    return Complex(a.re-b.re, a.im-b.im);
```

```
}
```

Class Basics – member/static *(folder 3)*

```
class List
{
public:
    static int getMaxSize();
    int getSize();
    // static int max_size=1000; //error! (declare outside)
    int size=0;
};
```

```
int List::max_size=1000; //ok, in one cpp file
```

```
int main()
{
    List l;
    l.getSize();
    List::getMaxSize();
    l.getMaxSize(); //compiles ok, but bad style
}
```

this

```
static int List::getMaxSize() //no this!
{
    return this->size; // compile error!
    return max_size; // ok
}
int List::getSize()
{
    return this->size; //ok
}
```

What file-names should we use?

- The C++ compiler does not care how your files are called.
- It is common to put a class declaration in file `ClassName.hpp` (or `ClassName.h`) and the class implementation in file `ClassName.cpp`.
- **Why is it better?**
 - Hiding implementation details.
 - Saving compilation time – when you have a good **Makefile** (*see folder 4*).