

CH-230-A

# **Programming in C and C++**

C/C++

## **Lecture 10**

Dr. Kinga Lipskoch

Fall 2020

## New Header Files

- ▶ `stdlib.h` or `math.h` can still be included in C++, but `<cstdlib>` or `<cmath>` is preferred
- ▶ Functions are then put into the `std` namespace
- ▶ Header files explicitly created for C++

# Namespaces

- ▶ C has only one global namespace
- ▶ Name collisions avoided by using prefixes
  - ▶ `jpeg_xxx`
- ▶ C++: `using namespace std;`
- ▶ Standard C++ libraries are all inside of the `std` namespace


## structs and classes


```
1 struct article {           // the C way
2     int id;
3     float price;
4 };
5 int add_article(struct article*, int id, float price);
6 ...
7 struct article a;
8 add_article(&a, 1234, 9.99);
```

```
1 class Article {           // the C++ way
2     int id;
3     float price;
4     int add(float id, int price);
5 };
6 Article s;
7 s.add(1234, 9.99);
```

You can combine the data  
and the functionality

## The string Class (1)

- ▶ `string` is another class provided by the standard C++ library
- ▶ It handles a sequence of characters which may dynamically grow or shrink 
- ▶ Strings can be created in different ways

```
1 string empty;           // empty string
2 string a("this is also a string"); 
3 string b = "also this one";
4 empty = a;              // now they hold the same
5 empty += " 8";          // appending to a string
```

## The string Class (2)



- ▶ The `string` class has many methods performing useful operations
  - ▶ Appending, inserting, removing, concatenating, replacing, searching, comparing and more
- ▶ We are not covering all of them on the slides
- ▶ See Chapter 5 in the *C++ Annotations* book or check operators and methods on [www.cplusplus.com](http://www.cplusplus.com)

## Example with Strings

```
string_tester.cpp
```

# Deficiencies of C Structures

`cstruct.c`

- ▶ Any function is able to read and also write the variables one and two
- ▶ Uncontrolled access to the account
- ▶ Clients are able to directly manipulate data
- ▶ No guarantee that access is done in the “right way”



## struct in C++

Member functions, methods are part of the struct itself

```
1 struct account {                                in C++ it can contain data AND functionality
2     char name[100];                             Difference from class is data protection
3     unsigned int no;
4     double balance;
5
6     // functions inside struct
7     void createAccount(const char *name, ....);
8     void deposit(double amount);
9     void drawout(double amount);
10    void transfer(struct account *to, double
        balance);
11 };
```

# How to Define New Classes

- ▶ The keyword `class` is used to define a new class
  - ▶ `struct` with methods
- ▶ Two other keywords used when defining classes:
  - ▶ `private`: to define what is internal to the class
  - ▶ `public`: to define what can be used from outside the class
- ▶ There exists a third keyword, `protected`, which will be introduced when we will talk about `inheritance` in more detail

# Information Hiding



- ▶ While designing a class it is necessary to devise which information should be visible and which one should not be visible to class users
  - ▶ This choice has to be done for both data members and methods
- ▶ The visible (`public`) subset of data and methods is called the `interface` of the class

# Information Hiding: Why?

- ▶ **Protection:**
  - ▶ Users are not allowed to use class data not belonging to themselves (data integrity)
- ▶ **Modularity:**
  - ▶ An interface is a **contract** between the class developer and the class user
  - ▶ As long as the interface does not change, the private part of the class can be changed without the need to modify the code that uses that class

## private and public

- ▶ **General rule:** data should be kept private and methods should be provided to access (read and write) them
- ▶ There may be exceptions to this principle, mainly due to efficiency needs
- ▶ Methods providing functionality needed by class users will be `public`
- ▶ Methods used to implement these functionality should be `private`

# Critters

- ▶ Critter have several properties (name, color, hunger)
- ▶ Data concerning these properties will be kept private
- ▶ Methods should be provided to write those data (setter methods)
- ▶ A method to get data (e.g., name for sorting – getter method)
- ▶ An additional method can be to print the data to the screen

# Implementation of the class Critter

- ▶ It is common to split the coding into two components
  - ▶ A header file specifies how the class looks like, i.e., its data members and methods
    - ▶ Class declaration
  - ▶ A C++ file defines how the methods are actually implemented
    - ▶ Class definition
- ▶ `Critter.h`
- ▶ `Critter.cpp`

## Compile the class Critter

Critter.cpp can be compiled but:

- ▶ It is just a model (no instances up to now)
- ▶ No main function, so it is necessary to instruct the compiler to avoid the linking stage
- ▶ `g++ -Wall -c Critter.cpp` generates `Critter.o`



## A Test Program

- ▶ `testcritter.cpp`
- ▶ Putting all together:  
`g++ -c testcritter.cpp`  
`g++ testcritter.o Critter.o -o testcritter`
- ▶ Could also be done by just one command:  
`g++ testcritter.cpp Critter.cpp -o testcritter`
- ▶ Execute:  
`./testcritter`

## Some Comments on `testcritter.cpp`

An instance of type `Critter` has been created

- ▶ Classes define Abstract Data Types (ADT)
- ▶ Once defined, they are types as language defined types, so it is possible to pass them as parameters to functions, declare pointers to ADT, etc.

## How to Invoke Methods

- ▶ Public methods can be invoked by using the selection operator, which is a dot

```
Critter c;  
c.setName("Gremlin");
```

- ▶ Methods must be applied to instances and not to classes

```
Critter.setName("Gremlin"); // wrong!
```

- ▶ With the notable exception of static elements (to be covered later)
- ▶ Method invocation evokes procedure call

## How to Access Data Members

- ▶ With the selection operator it is also possible to access (read write) data members, provided they are accessible

```
Critter c;
```

```
c.name = "Bitey";    // wrong: private
```

- ▶ Note the similarity with the selection operator used to access a C `struct`

## Specifying the Definition of a Class

Methods defined in the header file are usually implemented in a different source file

- ▶ Include the header (the compiler needs to know the shape of a class before checking methods)
- ▶ When defining a method specify the name of the class it belongs to:

```
void Critter::setName(string name) { ... }
```

- ▶ There can be more methods called `setName` in different classes, so it is necessary to specify which one it is being defined

## Defining a Method

When implementing a method it is not necessary to use the selection operator to call methods of the same class or to access data members

- ▶ Access defaults to the local instance

```
1 void Critter::setName(string newname) {  
2     name = newname;  
3 }
```

## More on Methods

- ▶ There is a strong correspondence between method calls and function calls
- ▶ A method:
  - ▶ Can accept parameters
  - ▶ Returns a typed value to the caller
  - ▶ Can call functions and other methods, including itself
  - ▶ Can include iterative cycles, local variables declaration, etc.
- ▶ A method is allowed to access data members and other methods in its class

## Instances of the Same Class

- ▶ Instances of the same class have the same set of data, but they are replicated so that they do not overlap

```
1  Critter a, b;  
2  a.setHunger(1);  
3  b.setHunger(4);  
4  cout << a.getHunger() << " "  
5      << b.getHunger();
```

will print

1 4

- ▶ a and b have a different memory space, so their modifications are independent



## When Should Data Members be `public`?

- ▶ The interface of a class should be *minimal*
  - ▶ This gives least commitments in what you should keep untouched in order to avoid modifying client code
- ▶ Exceptions: if you need to access a data very frequently, the use of setter and getter methods may result in a bottleneck (after all it is a function call)
- ▶ In those cases you could consider to make a data member `public` (but you can also declare the method as `inline`)