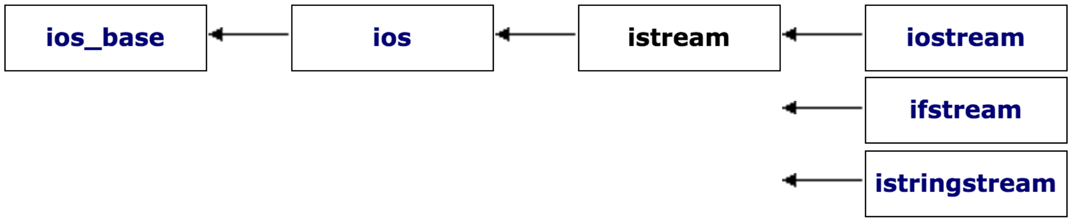
## 4 Inheritance

- Inheritance refers to derived classes
  - Derived classes are obtained from another class by adding features
  - The class of input-file streams is derived from the class of all input streams by adding member functions such as open and close
  - **cin** belongs to the class of all input streams, but not the class of input-file streams

### 4.1 Inheritance and Streams

- **cin** and an input-file stream are input streams
  - Input-file streams are members of the class **ifstream**
    - Can be connected to a file
  - **cin** is a member of the class **istream (no 'f ')**
    - Cannot be connected to a file
  - The **ifstream** class is a derived class of the **istream** class



`http://www.cplusplus.com/reference/istream/istream/`

#### 4.1.1 Stream Parameters

- Example:

```
void two_sum(ifstream& source_file)
{
        int n1, n2;
        source_file >> n1 >> n2;
        cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}
```

- This code could be called using

```
ifstream fin;
fin.open("input.dat");
two_sum (fin);
```

- Suppose you wished to use function **two_sum** with **cin**
- Since **cin** and input-file streams are both input streams, this call to two_sum seems to make sense:

```
two_sum(cin);
```

  but it will not work!

- This version of **two_sum** works with **cin**:

```
void better_two_sum(istream& source_file)
{
        int n1, n2;
        source_file >> n1 >> n2;
        cout << n1 << " + " << n2 << " = " << (n1 + n2) << endl;
}
```

- **better_two_sum** can be called with:

```
better_two_sum(cin);
```

### 4.1.2 Derived Classes and Parameters

- **better_two_sum** can also be called with:
```
ifstream fin;
fin.open("input.dat");
better_two_sum(fin);
```
- **fin** is of two types
  - **fin** is an input-file stream
  - **fin** is also of type **istream**
  - **fin** has all the features of the input stream class, plus added capabilities
- A formal parameter of type **istream** can be replaced by an argument of type **ifstream**

### 4.1.3 sample17.cpp

```
#include <iostream>
#include <fstream>

void better_two_sum(std::istream& source_file)
{
    int n1, n2;
    source_file >> n1 >> n2;
    std::cout << n1 << " + " << n2 << " = " << (n1 + n2) << std::endl;
}

int main(int argc, const char *argv[])
{
    // input from keyboard
    std::cout << "Input two integer numbers : ";
    better_two_sum(std::cin);

    // input from a file
    std::ifstream fin;
    fin.open("input.dat");
    better_two_sum(fin);
    fin.close();

    return 0;
}
```

### 4.1.4 Derived Class Arguments

- A restriction exists when using derived classes as arguments to functions
  - A formal parameter of type **istream**, can only use member functions of the **istream** class
  - Using an argument of type **ifstream** with a formal parameter of type **istream** does not allow using the open and close methods of the **ifstream** class!
    * Open files before calling the function
    * Close files after calling the function

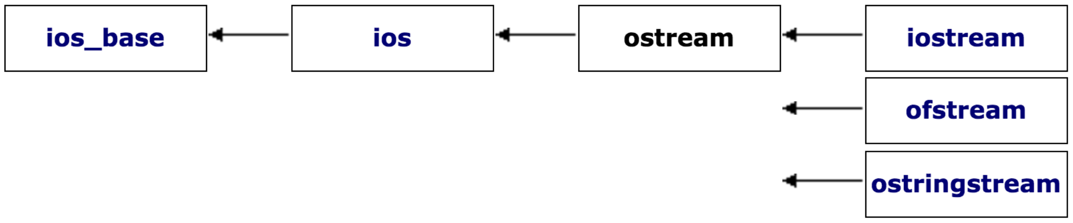
## 4.2 Inheritance Relationships

- If class B is derived from class A
  - Class B is a derived class of class A

- Class B is a child of class A
- Class A is the parent of class B
- Class B inherits the member functions of class A

## 4.3 Inheritance and Output

- **ostream** is the class of all output streams
  - **cout** is of type **ostream**
  - `http://www.cplusplus.com/reference/ostream/ostream/`



- **ofstream** is the class of output-file streams
  - The **ofstream** class is a child class of **ostream**
  - This function can be called with **ostream** or **ofstream** arguments

```
void say_hello(ostream& any_out_stream)
{
        any_out_stream << "Hello COSC3000/6000";
}
```

### 4.3.1 sample18.cpp

```
//
// sample18.cpp
//
#include <iostream>
#include <fstream>

void say_hello(std::ostream& any_out_stream)
{
    any_out_stream << "Hello COSC3000/6000\n";
}

int main(int argc, const char *argv[])
{
    // output to screen
    say_hello(std::cout);

    // output to a file
    std::ofstream fout;
    fout.open("output.dat");
    say_hello(fout);
    fout.close();

    return 0;
}
```

#### 4.3.2 Derived Class Arguments

- A restriction exists when using derived classes as arguments to functions
  - A formal parameter of type **ostream**, can only use member functions of the **ostream** class
  - Using an argument of type **ofstream** with a formal parameter of type **ostream** does not allow using the open and close methods of the **ofstream** class!
    - Open files before calling the function
    - Close files after calling the function

### 4.4 Default Arguments

- Wc can define a default value for input arguments
  - A default value can be specified in the parameter list
  - The default value is selected if no argument is available for the parameter
- The **say_hello** header can be written as

```
void say_hello(istream & in_stream = std::cout)
{
        any_out_stream << "Hello COSC3000/6000";
}
```

  - If **say_hello** is called without an argument, **cout** is used

### 4.5 Multiple Default Arguments

- When some formal parameters have default values and others do not
  - All formal parameters with default values must be at the end of the parameter list
  - The function call must provide at least as many arguments as there are parameters without default values

### 4.6 Default Argument Example

- 
```
void default_args(int arg1, int arg2 = -3)
{
        cout << arg1 << ' ' << arg2 << endl;
}
```
- default_args can be called with one or two parameters
- `default_args(5); //output is 5 -3`
- `default_args(5, 6); //output is 5 6`