Argument Manager Setup

To ensure that your program will interact with the server as intended, include:

1. “`#include ArgumentManager.h`”
2. Parameters “`int argc`“ and “`char *argv[]`“ in the main function

main.cpp

```cpp
#include "ArgumentManager.h"


int main(int argc, char *argv[]){
// CODE GOES HERE
}
```

Read / Write Using Argument Manager

To read input files and write output files, include the following lines (two ways)

input / Output streams must be formatted as shown to submit to Filezilla successfully

main.cpp (method 1)

```cpp
#include "ArgumentManager.h"
#include <fstream>
int main(int argc, char *argv[]){
    ArgumentManager am(argc, argv);

    string input = am.get("input");
    string output = am.get("output");

    ofstream ofs(output);
    ifstream ifs(input);
}
```

main.cpp (method 2) ← better

```cpp
#include "ArgumentManager.h"
#include <fstream>
int main(int argc, char *argv[]){
    ArgumentManager am(argc, argv);
    ifstream input(am.get("input"));
    ofstream output(am.get("output"));
}
```

Manual Local Testing With Argument Manager

To test locally, change "`input`" and "`output`" to your custom test cases.

Remember to reverse these changes when submitting to the server.

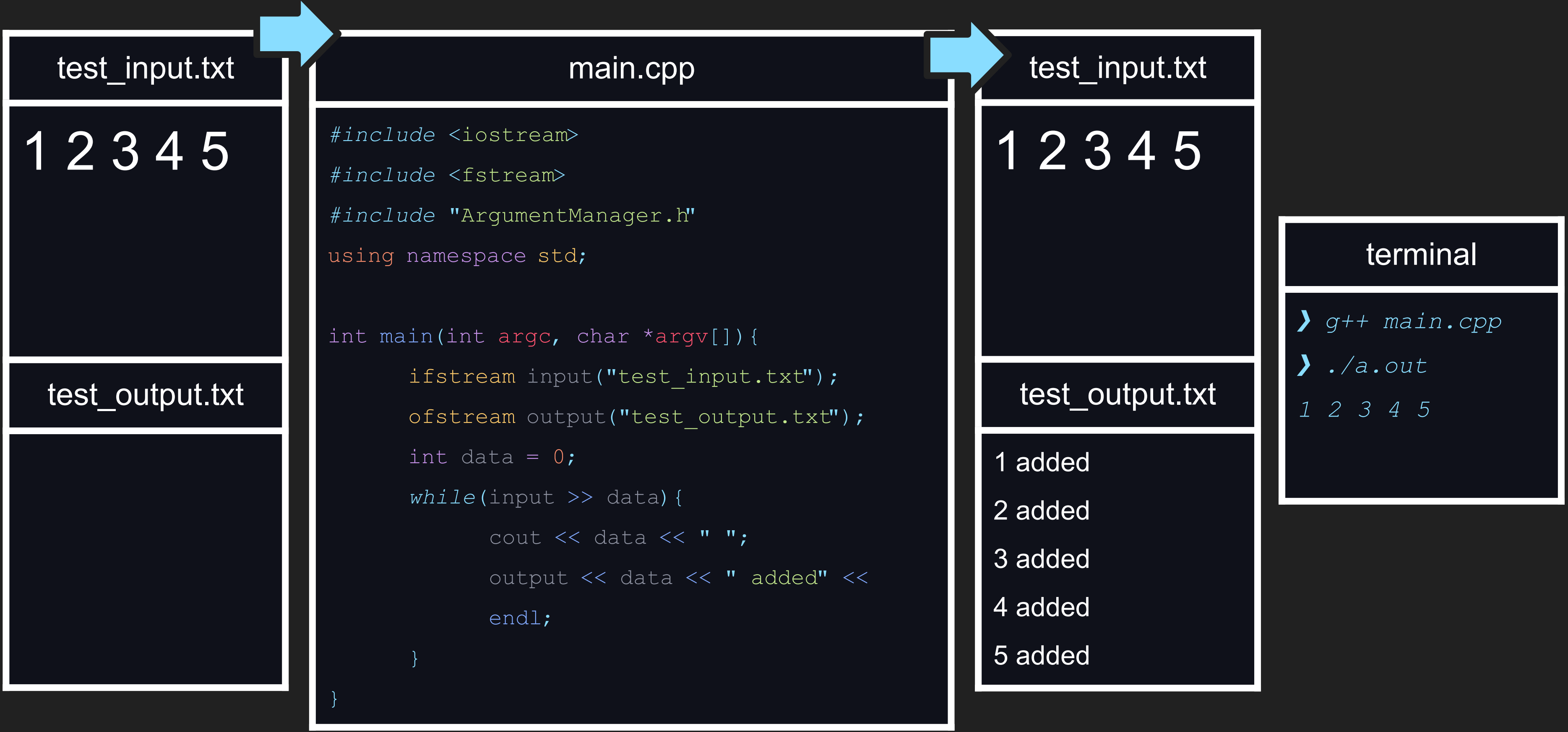
test_input.txt	test_output.txt
1 2 3 4 5	

main.cpp

```
#include <iostream>
#include <fstream>
#include "ArgumentManager.h"
using namespace std;

int main(int argc, char *argv[]){
    ifstream input("test_input.txt");
    ofstream output("test_output.txt");
}
// INPUT/OUTPUT MUST BE FORMATTED AS SHOWN BELOW TO
SUBMIT TO SERVER
// ArgumentManager am(argc, argv);
// ifstream input(am.get("input"));
// ofstream output(am.get("output"));
```

Manual Local Testing With Argument Manager Example



test_input.txt

```
1 2 3 4 5
```

test_output.txt

main.cpp

```cpp
#include <iostream>
#include <fstream>
#include "ArgumentManager.h"
using namespace std;

int main(int argc, char *argv[]){
	ifstream input("test_input.txt");
	ofstream output("test_output.txt");
	int data = 0;
	while(input >> data){
		cout << data << " ";
		output << data << " added" <<
		endl;
	}
}
```

test_input.txt

```
1 2 3 4 5
```

test_output.txt

```
1 added
2 added
3 added
4 added
5 added
```

terminal

```
❯ g++ main.cpp
❯ ./a.out
1 2 3 4 5
```

Command Example

Sometimes an assignment will require command files. Use them the same way as input files

input1.txt

```
1 2 3 4 5
```

command1.txt

```
2 2 3 3 4
```

main.cpp

```cpp
#include <iostream>
#include <fstream>
#include "ArgumentManager.h"
using namespace std;

int main(int argc, char *argv[]){
    ifstream input("input");
    ifstream command("command");
    ofstream output("output");

    int data = 0;
    int data2 = 0;
    while(input >> data){
        command >> data2;
        if(data == data2){
            output << data << " added" <<
            endl;
        }
    }
}
```

terminal

```
❯ sh test.sh
❯ Test case 1
PASSED.
```

output

```
2 added
3 added
```

Validate Passing Locally (1)

It is recommended to check if your program is passing the given test cases locally on your machine and on the server.

To check if your program's outputs match the given answer files ensure:

- You are using argument manager input/output
- Your program compiles and runs without errors
- There is an "test.sh" file in your directory
- Have the proper input.txt, ans.txt, and output.txt

main.cpp

```cpp
#include "ArgumentManager.h"
#include <fstream>
int main(int argc, char *argv[]){
        ArgumentManager am(argc, argv);
        ifstream input(am.get("input"));
        // CODE GOES HERE
        ofstream output(am.get("output"));
}
```

Validate Passing Locally (2)

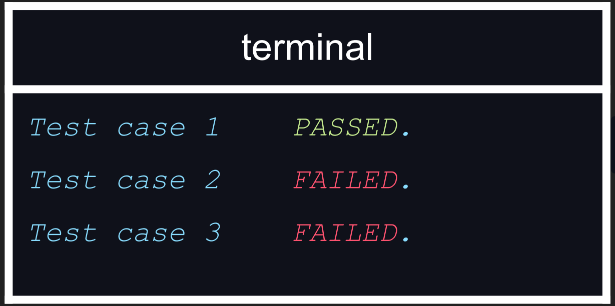
In your terminal/putty (macOS) or putty/git (Windows) ensure that you are in the correct directory.

You can navigate directories using terminal commands

- ❯ cd hw0 (changes directory to hw0)
- ❯ ls (lists files current in directory)
- ❯ cd .. (exit current directory)

Once you are ready to test your files run the command: ❯ sh test.sh OR ❯ ./test.sh

If done correctly, the program will output PASSED or FAILED for each test case in the console



```
Test case 1     PASSED.
Test case 2     FAILED.
Test case 3     FAILED.
```

In this example your program, when given input1.txt, produces the correct output that matches with ans1.txt. The output for input2.txt and input3.txt does not match with their corresponding ans files

Validate Passing on Server

Connect to the server by logging into Putty/Terminal with your given credentials

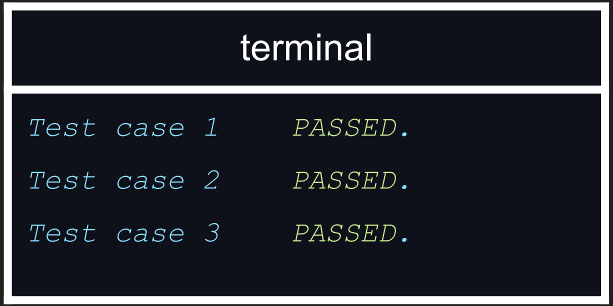
Connect to fileZilla server using your given credentials

Upload your assignment folder to the server, ensure that your directory is named correctly (lowercase & no spaces). Include .cpp, ArgumentManager.h, testcase (no output.txt files), and test.sh files.

Navigate to the directory and run the program

1. ❯ cd *foldername*
2. ❯ chmod u+x test.sh
3. ❯ sh test.sh

Delete all .txt files after validating passing cases →



terminal

```
Test case 1     PASSED.
Test case 2     PASSED.
Test case 3     PASSED.
```

Run ❯ chmod -R 755 *assignment#* in terminal at root directory

Logout and close Filezilla & Putty/Terminal

.stdout, .diff files, & GRADE files

A grading script will be run at midnight of the due date. **Ensure that everything is running correctly and is submitted to the server before this date.**

Once grading is complete, you will notice a new archive folder in your directory.

This folder will contain a .stdout, .diff, GRADE, and all the additional test case files.

- .stdout → contains your program's output for each test case
- .diff → contains errors, which test cases failed
- GRADE → run ❯ cat GRADE in terminal/putty to view your grade

Use the additional test case files to fix your program for the regrade (if eligible)

Additional Commands

A simple list of basic commands that you can run in PuTTY (Terminal) to navigate the server and test your code:

- mkdir “folder name” - make a folder
- rm -r “folder name” - remove a folder
- rm “file name” - remove a file
- ls - List of all the files in that folder
- cd “folder name” - Used to Enter a specific file in that folder
- cd .. - Used this to exit out of the folder you are currently in
- cat “filename” - see all contents in a file
- chmod u+x test.sh - give you permission to run the test file
- ./test.sh - runs the test file to test your code