

Writing a simple program

PHYS2G03

© James Wadsley,
McMaster University

hello.cpp

A basic C++ source file

```
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}
```

Programming: Key Elements

In order of importance:

1. Designing the Program
2. Testing the Program
3. Writing the Program

Note: Writing has the strongest dependence on the actual Programming Language used

Actual steps:

- 1. Designing the Program**
- 2. Writing the Program**
- 3. Testing the Program**
- 4. Writing the Program**
- 5. Testing the Program**

...

Testing is an ongoing process

Designing a program

1. State the problem
2. Analyse the problem: Break it down into simple steps.

For each step:

Decide what each step entails – what data needs to be available, what new data comes out

Next: Begin writing

3. Generate the code to solve the problem, step by step

Example Code

calc.cpp

Example: Write a program

1. The problem

Write a program to add two integers together and report the answer. The user supplies 2 integers a and b.

calc Program

2. Analysis: Major Steps

- (1) Read in 2 integers (input)
- (2) Calculate the sum of the integers
- (3) Report the answer (output)

Top-down Structure Plan for whole program

Part (2) is trivial – all CPUs can add integers. The whole program is just one main function

Writing the calc program

3. Solution

The solution is a self-contained piece of code:

e.g. calc.cpp

```
cp -r /home/2G03/calc ~/
cd ~/calc
```

```
gedit calc.cpp &
c++ calc.cpp -o calc    (OR make calc)
calc
```


Calc Program calc.cpp

```
#include <iostream>
```

```
int main()
```

```
{
```

```
}
```

This is a bare bones program

I have assumed that we will print something so I **included iostream** but its otherwise a skeleton -- could do anything

iostream gives us
std::cout and std::cin

Calc Program calc.cpp

```
#include <iostream>

int main()
{
    // Calc program
    // Takes two integers, sums them and reports
    // the answer

    // 1. Input 2 integers

    // 2. Sum a and b

    // 3. Output the results
}
```

My usual next step is to use comments to make markers for where I expect to put code

It helps connect the plan on paper to the developing code

Step 1. requires input ...

Input/Output

For Output in C++, cout is able to work out what variables expect from their type

- int integer
- float real number

```
int x;
```

```
std::cout << x;      Print the value of x to screen
```

This is guaranteed to work if x is a standard type of variable (like an integer)

You can think of std::cout as the terminal (screen) and each << sends another item to the terminal

std::cout is not defined without #include <iostream>

Input/Output

For Input in C++, cin is able to work out what variables expect from their type

- int integer
- float real number

```
int x;
```

```
std::cin >> x;
```

Take what the user types and try to interpret it as an integer

Only works if user types an integer

You can think of std::cin as user input typed to the terminal. Each >> takes one more item from the terminal and tries to store it in the variable provided. It will just wait until the user types enough items.


Calc Program calc.cpp

```
#include <iostream>
int main()
{
    // Calc program
    // Takes two integers, sums them and reports the answer
    int a,b,c;

    // 1. Input 2 integers

    // 2. Sum a and b

    // 3. Output the results
}
```



We need somewhere to store integers
In C/C++ you *must* declare variables before you use them

Calc Program calc.cpp

```
#include <iostream>
int main()
{
    // Calc program
    // Takes two integers, sums them and reports the answer
    int a,b,c;

    // 1. Input 2 integers
    std::cout << "Please input two integers\n";
    std::cin >> a >> b;

    // 2. Sum a and b

    // 3. Output the results
}
```

We use `std::cout` to tell the user what we want

We use `std::cin` to get input from the terminal

The program will wait here until two integers are typed


Calc Program calc.cpp

```
#include <iostream>
int main()
{
    // Calc program
    // Takes two integers, sums them and reports the answer
    int a,b,c;

    // 1. Input 2 integers
    std::cout << "Please input two integers\n";
    std::cin >> a >> b;

    // 2. Sum a and b
    c = a + b;

    // 3. Output the results
}
```



Here we “do the work”
It is a trivial example in
this case to add $a+b$

Note: we store the
answer by assigning the
value of $a+b$ to be
stored in c
This is what $=$ means in
programming


Calc Program calc.cpp

```
#include <iostream>
int main()
{
    // Calc program
    // Takes two integers, sums them and reports the answer
    int a,b,c;

    // 1. Input 2 integers
    std::cout << "Please input two integers\n";
    std::cin >> a >> b;

    // 2. Sum a and b
    c = a + b;

    // 3. Output the results
    std::cout << "The sum of " << a << " and " << b << " equals " <<
        c << "\n";
}
```



Here we report to the user what answer is Using std::cout of the value c

There is added text around the answer

Calc Program calc.cpp

```
#include <iostream>
int main()
{
    // Calc program
    // Takes two integers, sums them and reports the answer
    int a,b,c;

    // 1. Input 2 integers
    std::cout << "Please input two integers\n";
    std::cin >> a >> b;

    // 2. Sum a and b
    c = a + b;

    // 3. Output the results
    std::cout << "The sum of " << a << " and " << b << " equals " <<
        c << "\n";
    return 0; // success;
}
```

Finally, good practice is
to tell the operating
system we ran
successfully

return 0;

calc Program

- Testing it out:

c++ calc.cpp -o calc

calc

make

calc

Compile by hand

run the program (if the compile worked)

The Makefile already knows how to make a program called calc by compiling calc.cpp (take a look at the Makefile)

```
[wadsley@phys-ugrad ~/calc]$ more Makefile
```

```
calc: calc.o
```

```
    c++ calc.o -o calc
```

```
calc.o: calc.cpp
```

```
    c++ calc.cpp -c
```

make calc

```
[wadsley@phys-ugrad ~/calc]$ make calc
c++ calc.cpp -c
c++ calc.o -o calc
[wadsley@phys-ugrad ~/calc]$ make
make: `calc' is up to date.
[wadsley@phys-ugrad ~/calc]$ gedit calc.cpp &
[1] 17994
[wadsley@phys-ugrad ~/calc]$ make
c++ calc.cpp -c
c++ calc.o -o calc
[wadsley@phys-ugrad ~/calc]$
```

Calc Program in action

```
[wadsley@phys-ugrad ~/calc]$ g++ calc.cpp -o calc
```

```
[wadsley@phys-ugrad ~/calc]$ ./calc
```

Please input two integers

2 2

The sum of 2 and 2 equals 4

```
[wadsley@phys-ugrad ~/calc]$ ./calc
```

Please input two integers

3

6

The sum of 3 and 6 equals 9

Abusing Calc Program

```
[wadsley@phys-ugrad ~/calc]$ calc
```

```
Please input two integers
```

```
a
```

```
The sum of 0 and 0 equals 0
```

```
[wadsley@phys-ugrad ~/calc]$ calc
```

```
Please input two integers
```

```
1.2 2
```

```
The sum of 1 and 0 equals 1
```

```
[wadsley@phys-ugrad ~/calc]$
```