

ARGUMENTS TO MAIN() WORKING WITH FRACTIONS LOOPS

Problem Solving with Computers-I

C++

```
#include <iostream>
using namespace std;

int main() {
    cout<<"Hola Facebook\n";
    return 0;
}
```



Let's code Fizzbuzz -1.0

\$ Enter a number: 1

1

\$ Enter a number: 2

2

\$ Enter a number: 3

fizz

\$ Enter a number: 4

4

\$Enter a number: 5

5

\$Enter a number: 6

fizz

\$Enter a number: 7

7

\$Enter a number: 15

fizz

Let's code Fizzbuzz -2.0 (taking arguments from main)

```
$ ./fizzbuzz 1
```

```
1
```

```
$ ./fizzbuzz 9
```

```
Fizz
```

```
$ ./fizzbuzz 15
```

```
Fizzbuzz
```

Passing arguments to main (via the command line)

- We can pass information into a C++ program through the command line when executing the program.
- The main function will need to have the following:

```
int main(int argc, char* argv[ ])
```

- ``int argc`` is the number of "arguments" the program has, including the executable name.
- ``char* argv[]`` is the "list" of arguments passed into the program.
 - `argv[0]`: name of the program
 - `argv[1]`: 1st argument, remember this is a C-string
 - Use `atoi` to convert a C-string to a number `atoi(argv[1])`

C++ types in expressions

```
int i = 10;
```

```
double sum = 1/i;
```

What is printed by the above code?

A. 0

B. 0.1

C. 1

D. None of the above

Formatting output to terminal

See pages 91 and 190 of textbook

```
int i =10;
double j = 1/static_cast<double>(i);
cout.setf(ios::fixed);      // Using a fixed point representation
cout.setf(ios::showpoint); //Show the decimal point
cout.precision(3);
cout<<j;
```

What is printed by the above code?

- A. 0
- B. 0.1
- C. 0.10
- D. 0.100
- E. None of the above

C++ for loops

For loop is used to repeat code (usually a fixed number of times)

General syntax of a for loop:

```
for (INITIALIZATION; BOOLEAN_EXPRESSION; UPDATE) {  
    // code  
    // ...  
}
```

1. Execute the INITIALIZATION statement.
2. Check if BOOLEAN_EXPRESSION is true.
 - * if true, execute code in the loop.
 - * execute UPDATE statement.
 - * Go back to 2.
 - * if false, do not execute code in the loop.
 - * exit the loop and resume program execution.

Continue and break

- `continue;`
 - can be used to stop the current iteration of a loop,
 - perform the UPDATE statement if necessary, re-check the `BOOLEAN_EXPRESSION`, and
 - continue with the next iteration of the loop.

* `break;` can be used to break out of the **current** loop and continue execution after the end of the loop.

```
for (int i = 0; i < 10; i++) {  
    if (i == 4)  
        continue;  
    if (i == 7)  
        break;  
    cout << "i = " << i << endl;  
}
```


The accumulator pattern

Write a program that calculates the series:

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n},$$

where `n` is specified by the user

While loops

A while loop is used to repeat code while some condition is true

```
while(BOOLEAN_EXPRESSION)
    //Code
}
```

Check if the `BOOLEAN_EXPRESSION` is true.

- * If true, the statements in loop will execute.
 - * at the end of the loop, go back to 1.
- * If false, the statements in the loop will not execute.
 - * the program execution after the loop continues.

do-while loops

A while loop is used to repeat code until some condition is no longer true

```
do{  
    // Code  
    // This code is executed at least once  
}while(BOOLEAN_EXPRESSION);
```

1. Execute the code in the loop
2. Check if BOOLEAN_EXPRESSION is true.
 - * If true, then go back to 1.
 - * If false, then exit the loop and resume program execution.

Nested for loops – ASCII art!

Write a program that draws a square of a given width

```
./drawSquare 5
```

```
* * * * *  
* * * * *  
* * * * *  
* * * * *  
* * * * *
```

Draw a triangle

Which line of the drawSquare code
(show on the right) would you modify
to draw a right angled triangle

```
./drawTriangle 5
```

```
*
* *
* * *
* * * *
* * * * *
```

```
6   for(int j = 0; j < n; j++){ //A
7       for(int i=0; i < n; i++){ //B
8           cout<<"* "; //C
9       }
10      cout<<endl; //D
11  }
12  cout<<endl; //E
13
```

Infinite loops

```
for(int y=0;y<10;y--)  
    cout<<"Print forever\n";
```

```
int y=0;  
for(;;y++)  
    cout<<"Print forever\n";
```

```
int y=0;  
for(;y<10;);  
    y++;
```

```
int y=0;  
while(y<10)  
    cout<<"Print forever\n";
```

```
int y=0;  
while(y=2)  
    y++;
```

How is the pace of the class?

- A. Too fast
- B. Fast, but I am able to catch up once I do the labs
- C. Slow
- D. Too slow
- E. Its fine for me

Next time

- C++ functions and function call mechanics
- Variable scope (local vs. global)