Lecture 28

Finishing a few things

Admin

- HW4 back Tues/Wed
- Quiz 2 back Thur/Friday
- GSIs are working hard on PA.
- Practical back next week.

Admin

- Project B due this Friday
- HW5 will be posted today by 6pm
  - due on Tuesday the 29th at noon.
  - Answers posted by 2pm.
  - Will shoot for 2 hours.
- Exam on Wednesday the 30th
  - Evening, rooms posted shortly.
- After the exam, mostly done with C++ in class
  - There will be about 2 C++ lectures and 5 Matlab lectures.
  - Project C (C++), HW6 (Matlab) a practical (Matlab) and the Final is all that will be left.

Today: Misc. Day

- Finish up time class example
- A few misc. C++ topics
- Drag reviewed
- Starting on Stacks

Time class

```
void Time::add(Time a, Time b)
{
  set(a.hour+b.hour, a.min+b.min, a.sec+b.sec);
}
```

- Key thing to note:
  - A function member of a class can access private members of other instances of ***that same class***.
- Thoughts:
  - How does this get called?
  - Change it so that the following call would work:
    - C=A.add(B) (C=A+B)

A few useful C++ topics

- Functions
  - Functions show up in 3 places:
    - In your program when you *use* them
      - Called "function call" or "function invocation"
    - In your program when you *write* them
      - Called "function definition"
      - The first line of the definition is called the "function header" or "function heading"
    - At the top of the program by itself
      - Called the "function declaration" or "function prototype"

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

int factorial (int value);          <-- Function declaration

main()
{
    int max, a;
    a=factorial(max);               <-- Function call
    cout << max << " factorial is equal to " << a << endl

}

int factorial (int value)           <-- Function header
{
    int i=1;
    int fact=1;

    while(i<value)
    {
        i++;
        fact=fact*i;
    }
    return(fact);
}
```

Function body

Function definition

A factoid

- Function definitions or declarations must come before a function call.

Pass by reference (1/3)

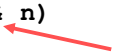
- Recall that in the following program:

```
int F(int n)
{
   n=n+4;
   return(n+4);
}
main()
{
   int a;
   int b=0;
   a=F(b);
}
```

  - b's value is zero at the end of the program
  - Argument's values don't change due to a function

Pass by reference (2/3)

```
int F(int & n)
{
   n=n+4;
   return(n+4);
}
main()
{
   int a;
   int b=0;
   a=F(b);
}
```



- Here, the value of "b" in the main will be 4.

Pass by reference (3/3)

- It is occasionally useful to be able to change an argument's value.
  - It is also very very common to overuse this feature.
  - In general, you should avoid using this unless
    - The data is truly being passed in and passed out.
    - If the program is really using the value as passed in AND needs to change the value, this can be a reasonable thing to do.
  - In C++ it is impossible to tell from the function call if a given argument is being passed by reference (rather than "pass by value"
    - A poor language feature as it makes the code harder to read.

RPN

- Consider writing a program which acts as a calculator…