1. So far

2. What is a class

3. A simple class

4. The structure of class
The story so far …

- We have seen various control flows.
- We have seen multi-dimensional arrays and the `char` data type.
- We saw the use of functions and calling methods.

This week…

Introduction to Classes
The basic objective of classes were

- to generalize type declarations such as `int`, `float` to user-defined types such as `poly`, `matrix`.

- to separate the user from the implementer. Thus, e.g., I may write `polynomials` class and various operations on it such as `evaluation`, `differentiation` etc. Any other user may use my polynomial definition. She must however use it only through the operations that I allow.
The basic objective of classes were

- to generalize type declarations such as `int, float` to user-defined types such as `poly, matrix`.
- to separate the user from the implementer. Thus, e.g., I may write `polynomials` class and various operations on it such as `evaluation, differentiation` etc. Any other user may use my polynomial definition. She must however use it only through the operations that I allow.

The **Class** framework is a special feature of C++.

- The class definition contains of two main chunks:
  - The **private** definition (or member variables),
  - The **public** operations.
- The **private** concerns details about the representation,
- The **public** defines operations which are exposed to the outside.
The `poly.cpp` class

```cpp
#include <iostream.h>
#include <math.h>
class poly
{
    private:
        float coefs[10];
        int degree;

    public:
        void ReadIn(void);
        int deg(void);
        float eval(float x);
        // poly diff(void);

    }; // poly

void poly::ReadIn(void)
{
    cin >> degree;
    if (degree > 9)
        cout << "degree bounded by 9";
    for (int i=0; i<=degree; i=i+1)
        cin >> coefs[i];
}

int poly::deg(void)
{
    return degree;
}

float poly::eval(float x)
{
    float r;
    r=0;
    for (int i=0; i<=degree; i=i+1)
        r=r+pow(x,i)*coefs[i];
    return r;
}
```
The poly.cpp class

class poly
{
    public:
        void ReadIn(void);
        int deg(void);
        float eval(float);
     //     poly diff(void);
};

int main()
{
    poly p;
    p.ReadIn();
    cout << p.eval(1.0);
}

Thus p.ReadIn() causes the construction of the polynomial

\[ p = 1 + 2x + 3x^2 + 4x^3 \]

The next statement

p.eval(1.0) evaluates it at \( x = 1 \).
The class structure

The basic structure of a program using classes is as follows:

```cpp
#include ...
class classname {
  private ...;
  public ...;
};

void classname::function {
  body
}
```

The class definition has two parts and must be done before the main program begins.

- **The Declaration**, which declare the class name, the private or **member** variables and the **methods**, which are the public ways of accessing objects.

- **The Methods**, which are the definition of the functions which operate on the local variables. These are the only methods by which an outer program may access the member variables.
The basic structure of a program using classes is as follows:

```cpp
#include ...

class classname
{
    private ...

    public ...
};

void classname::function
{
    body
}
```

The main uses of classes are:

- As opposed to `struct`, a `class` allows not only data but procedures which manipulate this data.
- This allows a separation: the person who writes the class may be different from the one who uses it.
- Classes allow us to remember data: coefs read only once, but polynomial evaluated repeatedly.
The Newton-Raphson technique:

- Use the function value and its derivative to compute the next candidate.
Root finding again

The Newton-Raphson technique:

- Use the function value and its derivative to compute the next candidate.

\[ x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \]
Root finding again

The Newton-Raphson technique:

- Use the function value and its derivative to compute the next candidate.

\[ x_{i+1} = x_i - \left( \frac{f(x_i)}{f'(x_i)} \right). \]

- We will upgrade our `poly` class to include differentiation.
newraph.cpp

class poly
{
    private:
        float coefs[10];
        int    degree;
    public:
        void ReadIn(void);
        int    deg(void);
        float eval(float);
        poly   diff(void);
};

This allows yet another public function called p.diff() on a polynomial p.
And here is the definition of \texttt{p.diff}.

\begin{verbatim}
poly poly::diff(void)
{
    poly q;
    q.degree=degree-1;
    for (int i=0; i<=q.degree; i=i+1)
        q.coefs[i] = (i+1)*coefs[i+1];
    return q;
}
\end{verbatim}

This allows yet another \texttt{public} function called \texttt{p.diff()} on a polynomial \texttt{p}. 
int main()
{
    poly p,q;

    p.ReadIn();
    q=p.diff();
    cin >> x >> tol;
    fval=p.eval(x);
    while ((fabs(fval)>tol)
        && (count<1000))
    {
        der=q.eval(x);
        x=x-fval/der;
        count=count+1;
        fval=p.eval(x);
    }

    cout << ...
int main()
{
    poly p,q;
    p.ReadIn();
    q=p.diff();
    cin >> x >> tol;
    fval=p.eval(x);
    while ((fabs(fval)>tol)
    && (count<1000))
    {
        der=q.eval(x);
        x=x-fval/der;
        count=count+1;
        fval=p.eval(x);
    }
    cout << ...
Whats the big deal?

2
-1  0  1
-1.5  0.0000000001

Thus $p = x^2 - 1$, the initial guess is $-1.5$ and the tolerance is $10^{-10}$!
Whats the big deal?

2
-1 0 1
-1.5 0.0000000001

Thus $p = x^2 - 1$, the initial guess is $-1.5$ and the tolerance is $10^{-10}$!

[sohoni@ns1-13]$ ./a.out <input
final root -1
iterations 4

Just 4 iterations.
What's the big deal?

2
-1 0 1
-1.5 0.0000000001

Thus \( p = x^2 - 1 \), the initial guess is \(-1.5\) and the tolerance is \(10^{-10}\)!

[sohoni@ns1-13]$ ./a.out <input
final root -1
iterations 4

Just 4 iterations.

3
0 -1 0 1
-1.5 0.0000000001

Thus \( p = x^3 - x \), the initial guess is \(-1.5\) and the tolerance is \(10^{-10}\)!
What's the big deal?

Thus $p = x^2 - 1$, the initial guess is $-1.5$ and the tolerance is $10^{-10}$!

```bash
[sohoni@nsl-13]$ ./a.out <input
final root -1
iterations 4
```

Just 4 iterations.

Thus $p = x^3 - x$, the initial guess is $-1.5$ and the tolerance is $10^{-10}$!

```bash
[sohoni@nsl-13]$ ./a.out <input
final root -1
iterations 5
```

5 iterations.
Assignment

- Write a function on poly to evaluate the $k$-th derivative at the point $x$.
- Use this function to write `p.multiplyby(q)` which multiplies polynomial $p$ by $q$ (and stores it in $p$).
- Now write `p.DivideBy(q)` and implement polynomial long division.