Lecture 02

C++ Programming Basics Review

- Object-Oriented Programming
- Dynamic Memory Allocation
- Template
- Inheritance
- Virtual Function
- Parallel Processing

- Object-Oriented Programming
  - Data Encapsulation
  - Inheritance
  - Polymorphism
- Building a wall around the data structure.

- The first step: Organizing your data.

Bouncing-Ball Example

- Without class:

```
const int nBall=10;
double m[nBall],x[nBall],y[nBall],vx[nBall],vy[nBall];
```

- Make a ball class:

```
class Ball
{
public:
        double x,y,vx,vy;
};
const int nBall=10;
Ball ball[nBall];
```

Defining behaviors of the Ball

- Class also can have member functions, which defines how the class can behave.

```
class Ball
{
public:
        double x,y,vx,vy;
        void Move(const double dt);
        void CalculateCollision(Ball &withThisBall);
        void BounceOnWall(void);
        void Draw(void) const;
};
```

Protecting member variables from accidental modification.

- C++ has a lot of mechanisms to catch and prevent programming errors.
- Some criticize C++ has too many features.
- You don’t have to use all of them. Start with what you understand and gradually expand your boundary.

- Protected and private members are one of such features.

Defining behaviors of the Ball

- Class also can have member functions, which defines how the class can behave.

```
class Ball
{
private:
        double x,y,vx,vy;
public:
        void SetPositionAndVelocity(double x,double y,double vx,double vy);
        void Move(const double dt);
        void CalculateCollision(Ball &withThisBall);
        void BounceOnWall(void);
        void Draw(void) const;
};
```

Template

- Templated n-dimensional vector class.

```
template <class T,const int N>
class VecN
{
protected:
        T v[N];
public:
        T Get(int n) const;
        void Set(int n,const T &incoming);
};
```

Operator overloading

- Defining math operators for the 2D vector class.

Dynamic Memory Allocation

- What if you don’t know the scale of the data that your program needs to deal with until run time?
- Solution: Dynamic memory allocation.
- Use new and delete operators.

A program that generates N non-repeating random numbers.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void Shuffle(int n,int dat[])
{
    for(int i=0; i<n; ++i)
    {
        int j=rand()%n;
        int a=dat[i];
        dat[i]=dat[j];
        dat[j]=a;
    }
}
```

```
int main(void)
{
    srand((int)time(nullptr));

    printf(">");

    char str[256];
    fgets(str,255,stdin);
    const int n=atoi(str);

    int *r=new int [n];
    for(int i=0; i<n; ++i)
    {
        r[i]=i;
    }
    Shuffle(n,r);

    for(int i=0; i<n; ++i)
    {
        printf("%d\n",r[i]);
    }

    delete [] r;

    return 0;
}
```

- Risk of memory leak
  - Cover the pointer with a class.
  - Use constructor and destructor to always delete an allocated array.
- Make it template and sub-class
- Make it copyable

Virtual Function

- Function integrator
- Make it run parallel