program: piece of code in any language
contains main()

O.S: Calls main()

application: .cc file containing main()

Class Particle   Particle.h   header file

Particle.cc   source file

```
double average(--); // declaration
double x;           // declaration
int Count();        // declaration
```

Particle.h

```
Class Particle {
    public:
        Particle (double, Vector3D);
        double momentum() const;
        Vector3D momentum() const;
        double mass() const {
            return --;
        }
};
```

declaration only

// implementation

Particle.cc

```
#include "Particle.h"
Particle::Particle(double E, double PZ) {

    ≡                       // implemetation

}

double   Particle::momentum() const {
           return 0;
}
```

~~gcc -c Particle.h~~

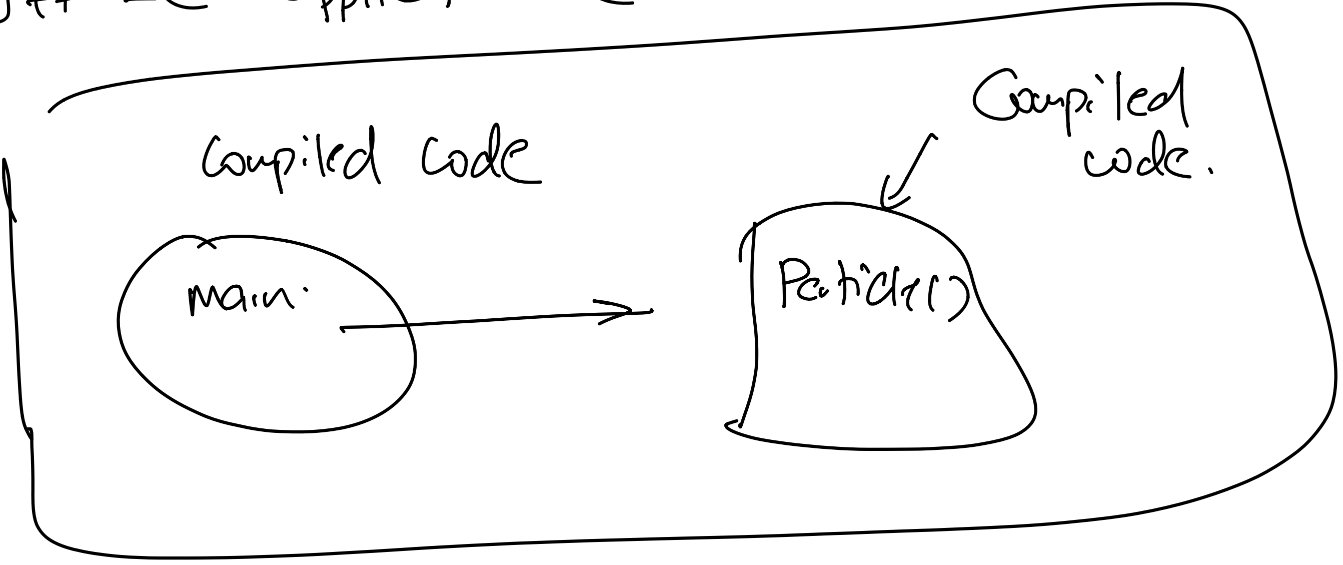
gcc -c Particle.cc

$\hookrightarrow$ Particle.o

application.cc

```
#include "Particle.h"
#include "vector3D.cc"        BAD ERROR
#include "vector3D.h"
int main() {
     Particle P(1.2, vector3D(1,2,3));
}
```

g++ -c applicaction.cc



Executable.: app

g++ -o app application.cc Particle.o

↳ application.o

.cpp
.cc
.cxx
.C

} C++ file

Particle.cc ⟶ g++ -c Particle.cc

↳ Particle.o

Jet.cc ——

Boson.cc ——

Detector.cc ——

Interaction.cc ——

Generator.cc ——

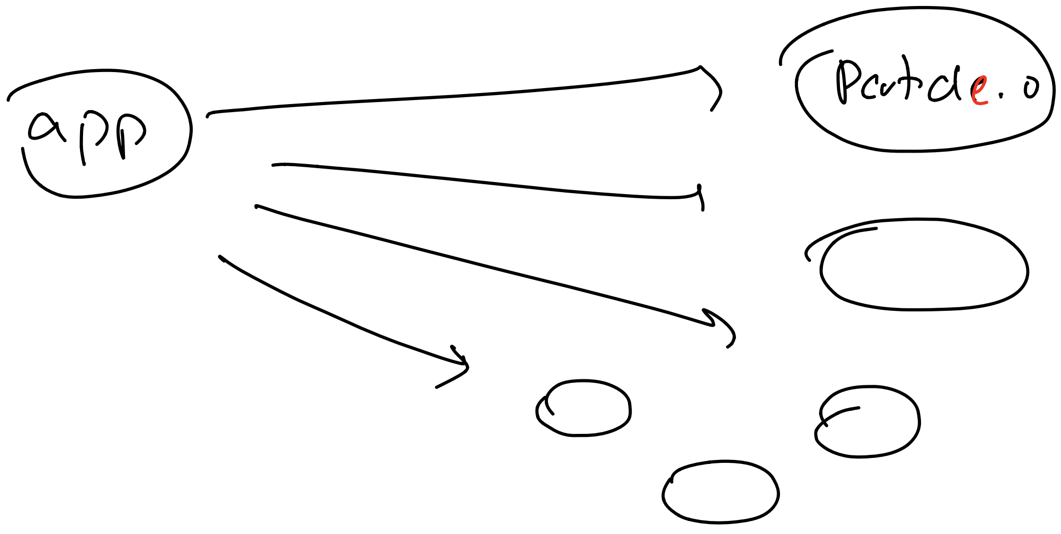
⋮

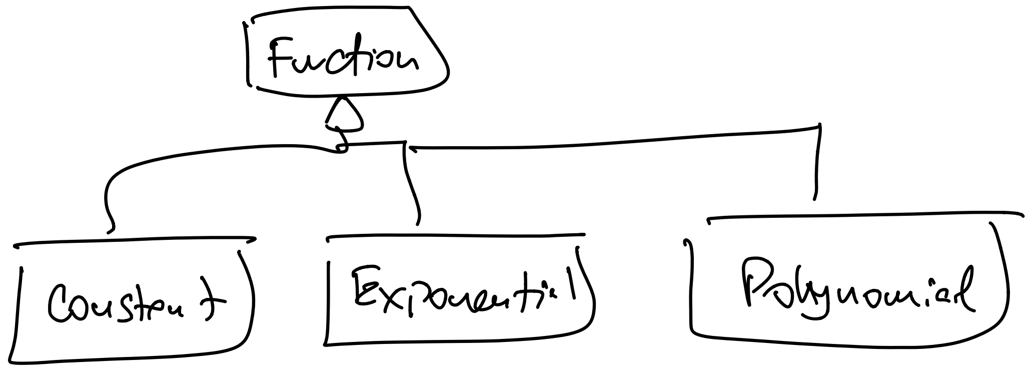
.o
⋮
.o

application.cc

```
#include Particle.cc
___
___
___
___
#include Generators.cc

int main() {
    cout << "hello" << endl;
    cout << " testing" << endl;
}
```

Compile only this





```
double   value(x)
double   derivative(x)
double   integrate(a,b)
```

```
double Function::value(double x) const {
        return 0;
}
```

Goal: have value() only for derived class
NOT FOR BASE CLASS

ABSTRACT Class

```
Function   f("f")   ← Compilation error:
~~f.value()~~?
```

```
Function* g = new Constant("c", 2.3);
```

↳ abstract class

derived/ Concrete class

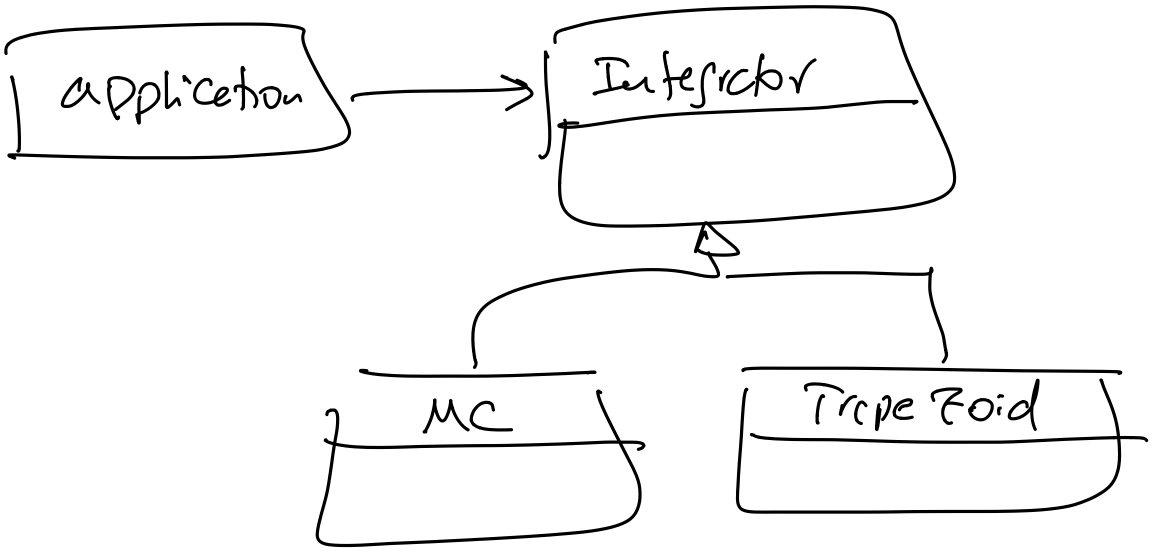
```
g = new Gauss("gauss", 0, 1.)
```

```
class Function {

    virtual double value(double) const = 0;
friend ostream& operator<<(ostream&, Function& ) ~~;~~

}
ostream& operator<<(ostream&, Function& );
```

not a member function

friend methods cannot be pure virtual

Strategy Pattern



```
Integrator* integ ;

   integ = new MCInteg( -- )
   integ = new TrapezIntegrator( -- )
```

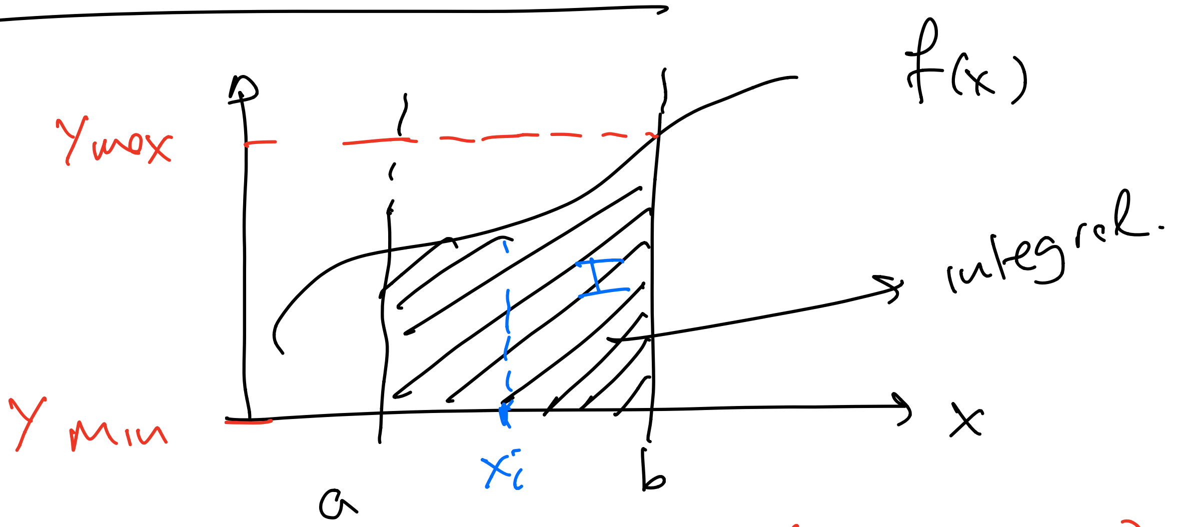
range: $x_1 \rightarrow x_2$ (two doubles)

steps: n integer

integrand: function

argument of integrate( )

Numerical MC integration



Area Rectangle $= (b-a) * (Y_{max} - Y_{min})$

random $x \in [a, b]$ $\quad x_i \quad f(x_i)$

random $y \in [Y_{min}, Y_{max}]$ $\quad y_i \quad y_i < f(x_i)$

$$\frac{I}{\text{Rectangle}} = \frac{N \text{ below Curve}}{N \text{ total}}$$