上海交通大學 SHANGHAI JIAO TONG UNIVERSITY | 李政道研究所 Tsung-Dao Lee Institute

C++ Training Week 1

Xu-Liang Zhu

TDLI

2021 Nov 21

Foreword

This cousre is

- for inpac cluster users
- basic knowledge to use and develop DarkSHINE Simulation Framework
- not complete at all

Reference

www.runoob.com/cplusplus

indico.cern.ch/event/979067

en.cppreference.com

Bjarne Stroustrup, The C++ Programming Language (4th Edition), *Addison-Wesley* 2013

Outlines

Why C++

Setup Remote Development Toolchains

C++ Basics

- Pointer
  - new expression
  - Smart pointers
- Class
  - Definition
  - Derived Class
  - Operator Overloading
  - Shallow Copy & Deep Copy

Modern C++

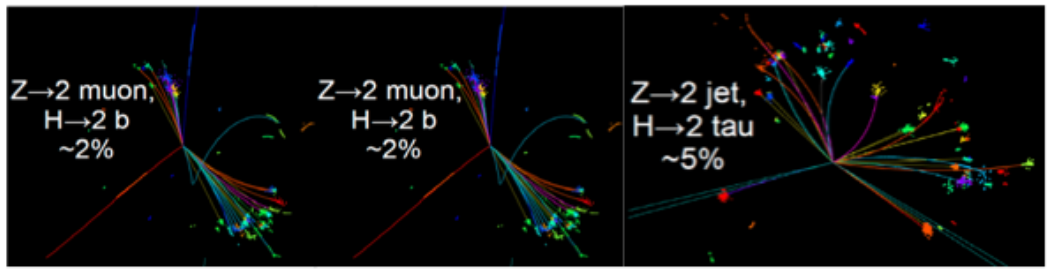
- STL
  - Containers
  - Algorithms
  - Iterators
- CMake

Collaborating in Git

Debugging

高能物理实验计算

- 大数据：多次测量的随机过程（多次独立实验）
  - 随机变量空间很大：产生的末态粒子极其丰富；
  - 精确测量需要大样本：大数据
- 大计算：末态的模式复杂（随机变量）
  - 参数估计：拟合及误差估计；
  - 物理图像还原非常复杂：图像处理、模式识别技术；等等



高能物理领域已经步入EB级的大数据时代

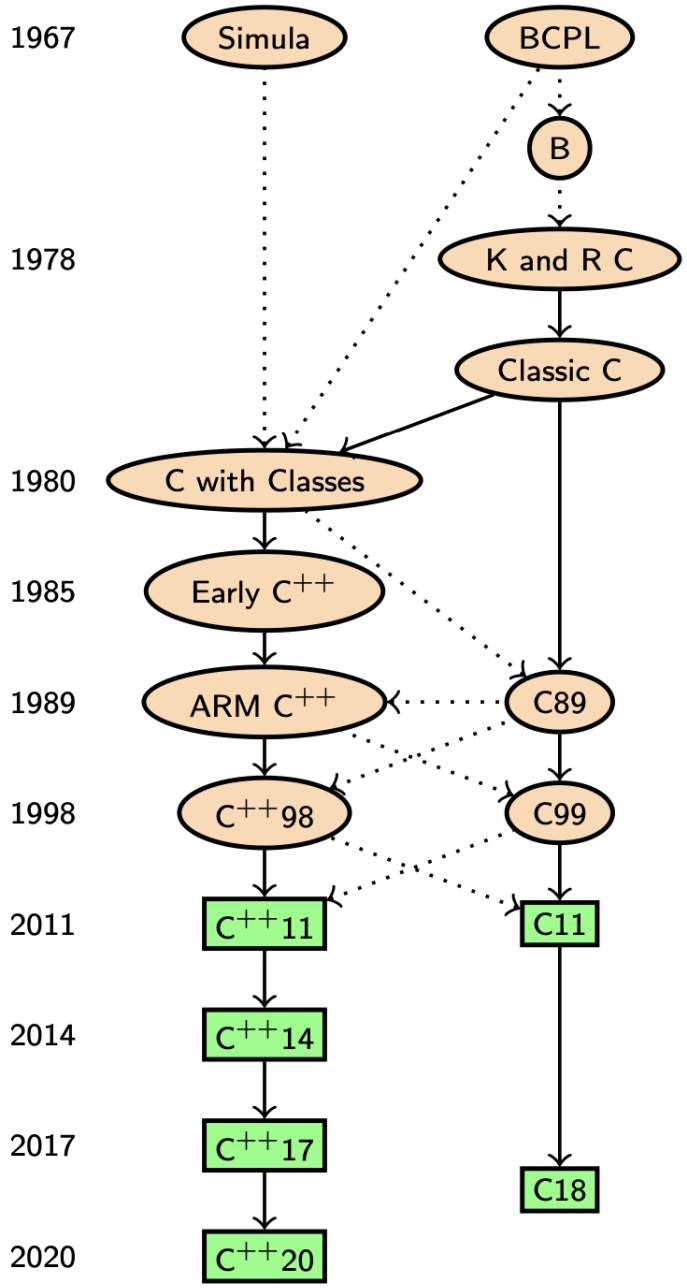
Why C++ our choice?

Adapted to large projects

- Strongly typed: What you see is what you get
- Object oriented: Maintainability, expandedability
- Widely used (and taught)
- Many available libraries: stl, boost, qt ...

Fast

- Compiled (unlike Python, Java or C#)
- Allows to go close to hardware when needed



C++ History

- Both C and C++ are born in Bell Labs
- C and C++ are still under development

How to use C++XX features

- Use a compatible compiler
- add e.g. `-std=c++17` compilation flags

C++	11	14	17	20
gcc	≥ 4.8	≥ 4.9	≥ 7.3	> 11
clang	≥ 3.3	≥ 3.4	≥ 5	> 12

Setup CLion

1. Install and activate CLion from

   https://lic.sjtu.edu.cn/Default/huatishow/tag/MDAwMDAwMDAwMLJ4iqE

2. Login bl-0 using any terminal.

3. copy file for environment variables

```
cp /lustre/collider/zhuxuliang/cpptrain.env ~
```

4. source this file when needed

```
source ~/cpptrain.env
```

5. find the path of compilers.

```
which cmake make gcc c++ gdb
```

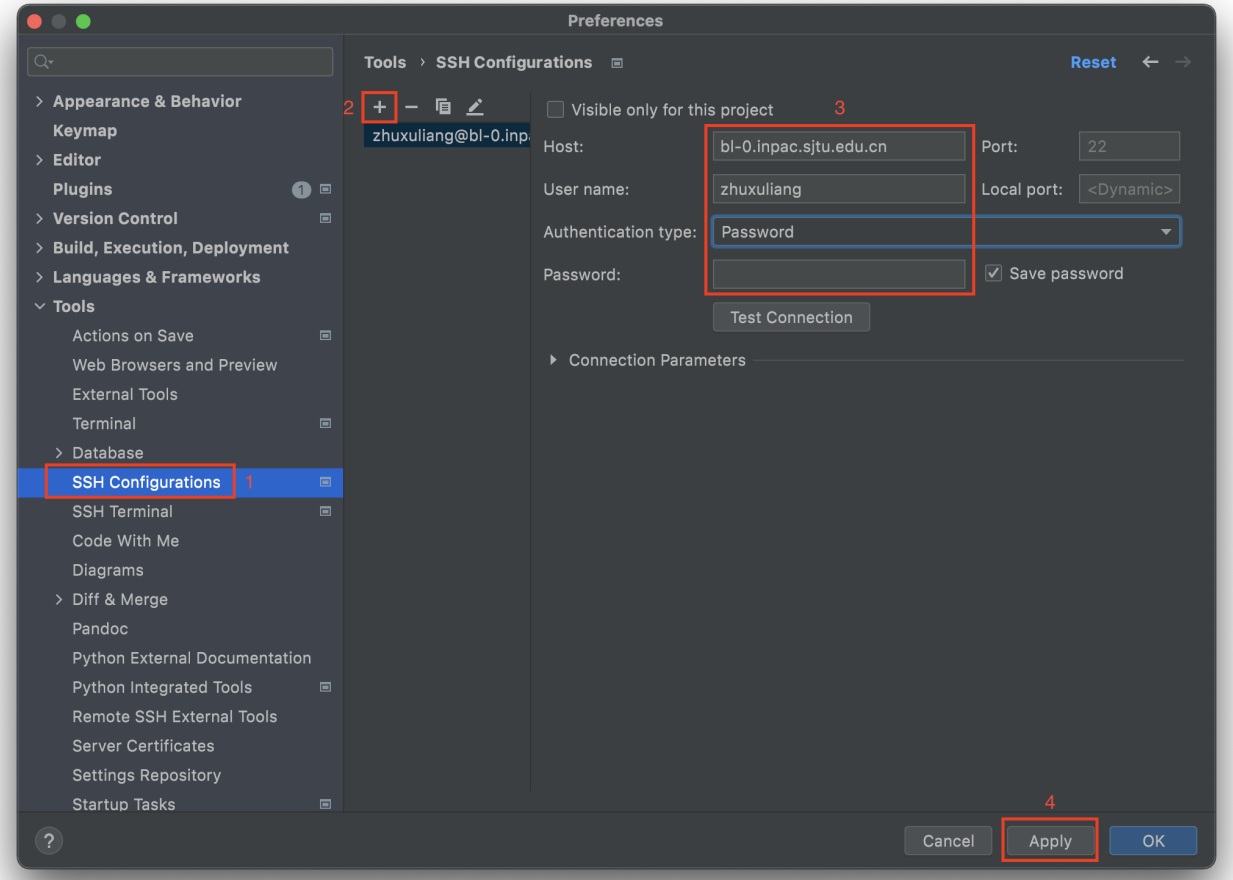
the output should be

```
/cvmfs/sft.cern.ch/lcg/views/LCG_97rc4python3/x86_64-centos7-gcc9-opt/bin/cmake
/usr/bin/make
/cvmfs/sft.cern.ch/lcg/releases/gcc/9.2.0-afc57/x86_64-centos7/bin/gcc
/cvmfs/sft.cern.ch/lcg/releases/gcc/9.2.0-afc57/x86_64-centos7/bin/c++
/cvmfs/sft.cern.ch/lcg/views/LCG_97rc4python3/x86_64-centos7-gcc9-opt/bin/gdb
```

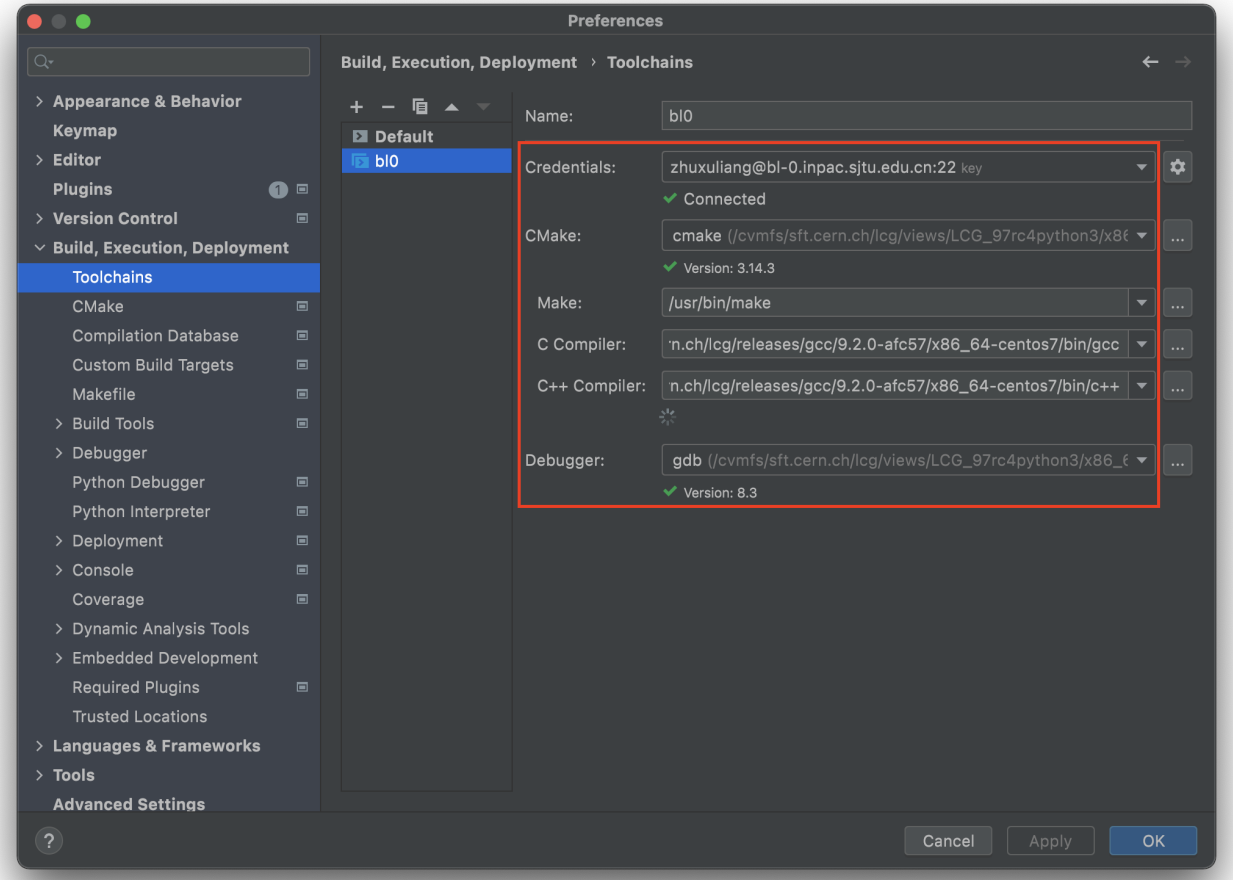
6. Copy some lines into `~/.bashrc` .in order to use c++17 features, and further.

```
cat /lustre/collider/zhuxuliang/forClion >> ~/.bashrc
cat ~/.bashrc # check
```

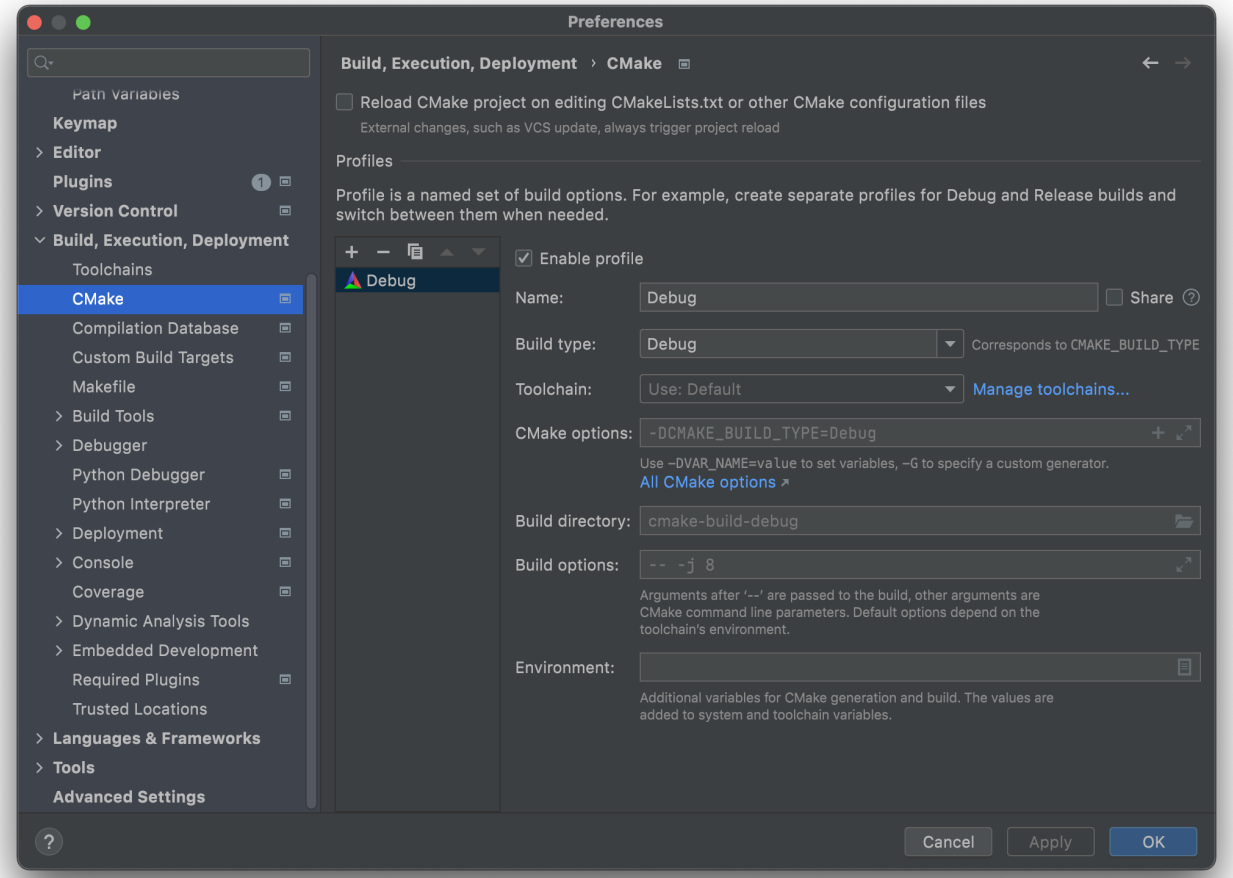
7. Configure SSH. Goto *Preference->Tools->SSH Configurations*, add the host `bl-0.inpac.sjtu.edu.cn` .



8. Configure Toolchans. Goto *Build, Execution, Deployment -> Toolchains*, add the remote host to the toolchain. Copy paths from step 5. into compiler paths.



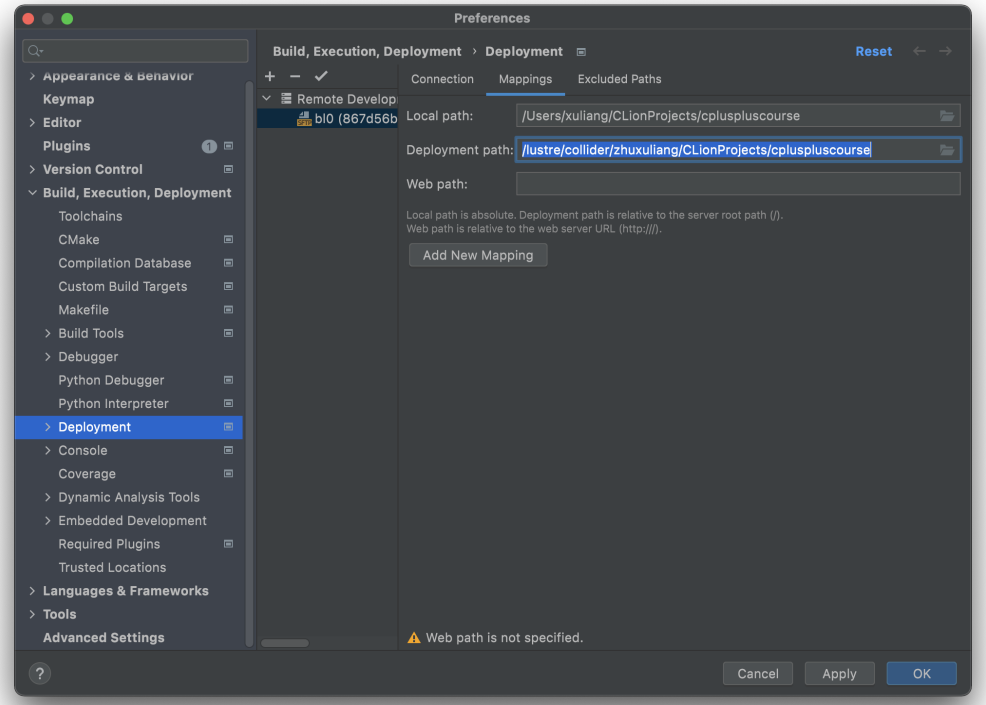
9. Configure CMake. Goto *CMake*, select the Toochain.



Get the training codes

1. Goto the git repository https://github.com/ykrsama/cpluspluscourse and copy the clone URL.
2. Open CLion, select **Get from VCS**, then paste the URL, wait for downlad to complete.

8. Modify the Deployment Mapping to your home. Goto *Deployment -> Mappings*, set *Deployment path* to `/lustre/collider/<USER>/CLionProjects/cpluspluscourse`. Change `<USER>` to your username.



hello

Just try compile and run hello.

C++ Basics

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
	0x3048
	0x3047
	0x3046
	0x3045
	0x3044
	0x3043
	0x3042
	0x3041
	0x3040

Pointers

```
int i = 4; // <-
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
	0x3048
	0x3047
	0x3046
	0x3045
	0x3044
	0x3043
	0x3042
	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i; // <-
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
	0x3048
	0x3047
	0x3046
	0x3045
	0x3044
	0x3043
	0x3042
pi=0x3040	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1; // <-

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
	0x3048
	0x3047
	0x3046
	0x3045
	0x3044
	0x3043
j=5	0x3042
pi=0x3040	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3}; // <-
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
	0x3048
	0x3047
ai=0x3043	0x3046
3	0x3045
2	0x3044
1	0x3043
j=5	0x3042
pi=0x3040	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai; // <-
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
	0x3048
pai=0x3043	0x3047
ai=0x3043	0x3046
3	0x3045
2	0x3044
1	0x3043
j=5	0x3042
pi=0x3040	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1; // <-
int k = *paj + 1;

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
	0x3049
paj=0x3044	0x3048
pai=0x3043	0x3047
ai=0x3043	0x3046
3	0x3045
2	0x3044
1	0x3043
j=5	0x3042
pi=0x3040	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1; // <-

// not compiling
int *pak = k;

// seg fault !
int *pak = (int*)k;
int l = *pak;
```

Memory layout	Address
	0x304A
k=3	0x3049
paj=0x3044	0x3048
pai=0x3043	0x3047
ai=0x3043	0x3046
3	0x3045
2	0x3044
1	0x3043
j=5	0x3042
pi=0x3040	0x3041
i=4	0x3040

Pointers

```
int i = 4;
int *pi = &i;
int j = *pi + 1;

int ai[] = {1,2,3};
int *pai = ai;
int *paj = pai + 1;
int k = *paj + 1;

// not compiling
//int *pak = k;

// seg fault !
int *pak = (int*)k; // <-
int l = *pak;
```

Memory layout	Address
?? pak=0x3	0x304A
k=3	0x3049
paj=0x3044	0x3048
pai=0x3043	0x3047
ai=0x3043	0x3046
3	0x3045
2	0x3044
1	0x3043
j=5	0x3042
pi=0x3040	0x3041
i=4	0x3040

new expression

Creates and initializes objects with dynamic storage duration, that is, objects whose lifetime is **not necessarily limited** by the scope in which they were created.

```
double* p = new double[]{1,2,3}; // creates an array of type double[3]
auto p = new auto('c'); // creates a single object of type char. p is a char*
```

Memory leaks

The objects created by new-expressions (objects with dynamic storage duration) persist until the pointer returned by the new-expression is used in a matching delete-expression. If the original value of pointer is lost, the object becomes unreachable and cannot be deallocated: a *memory leak* occurs.

This may happen if the pointer is assigned to:

```
void f()
{
    int* p = new int(7);
    p = nullptr; // memory leak
}
```

or if the pointer goes out of scope:

```
void f()
{
     int* p = new int(7);
} // memory leak
```

or due to exception

```
void f()
{
    int* p = new int(7);
    g();       // may throw
    delete p; // okay if no exception
} // memory leak if g() throws
```

Smat pointers

`std::shared_ptr` is a smart pointer that retains shared ownership of an object through a pointer. Several `shared_ptr` objects may own the same object. The object is destroyed and its memory deallocated when either of the following happens:

- the last remaining `shared_ptr` owning the object is destroyed;
- the last remaining `shared_ptr` owning the object is assigned another pointer via operator= or reset().

Exercise - pointers

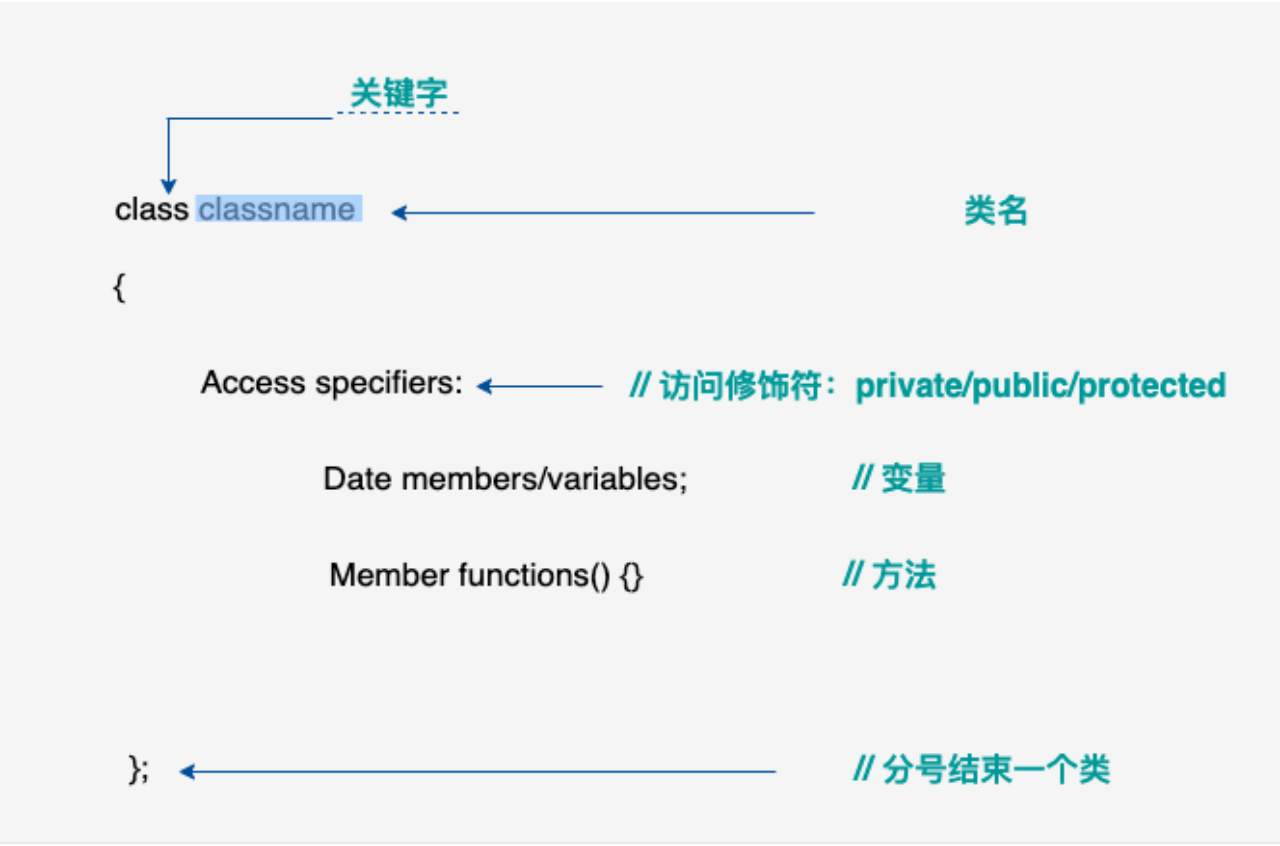
1. Compile and run pointers
2. Use delete to fix the memory leak?
3. Or, use shared_ptr to replace the raw pointer? (Please Google for how to use shared_ptr)

Class & Object

The aim of the C++ class concept is to provide the programmer with a tool for creating new types that can be used as conveniently as the built-in types.

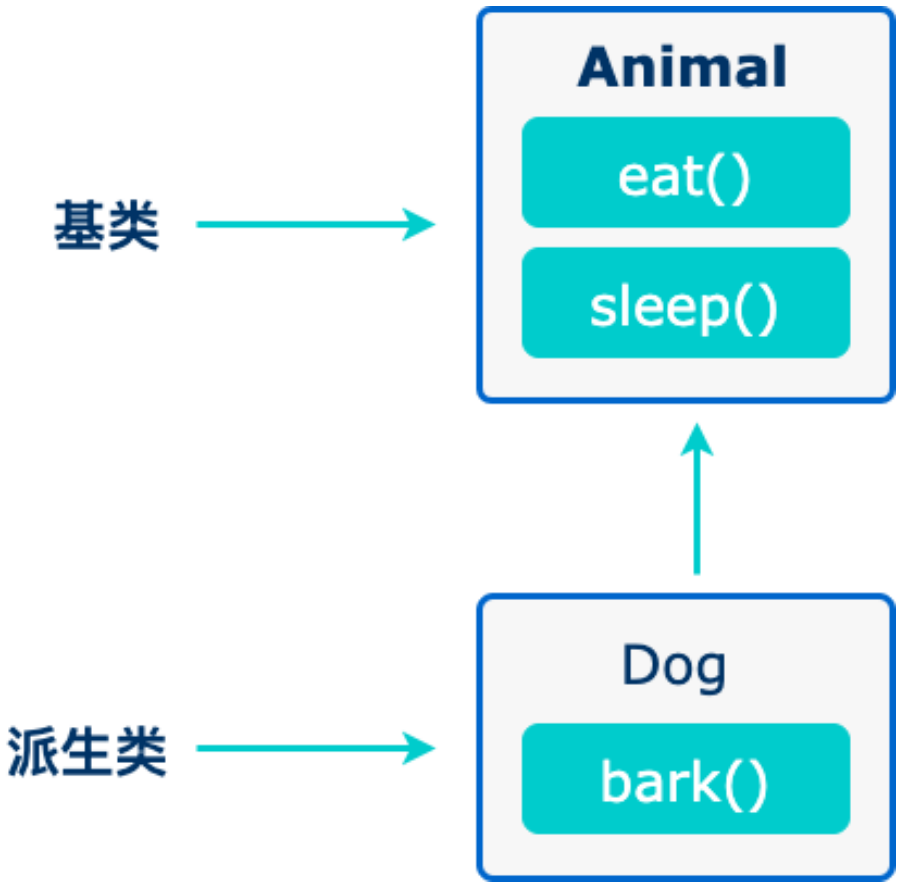
Example

```
// Define a Class
class Particle {
public:
    Particle(double m) { mass = m; }; // Constructer
    ~Particle(); // Destructer

    // getters
    double getMass() {return mass; };
    double* getMomentum() {return fourMomentum; };
    double getEnergy();

    // setters
    void setMomentum(double px, double py, double pz);

protected:
    double mass;
    double fourMomentum[4];
}

// Define Member Function
double Particle::getEnergy() {
    // ...
    return ...;
}

// Define Objects
Particle part1(0.5);
part1->setMomentum(0,0,1);
```



Inherit, Derived Class

```
// Base class
class Polygon {
public:
    Polygon(int n, float radius);
    float computePerimeter();
protected:
    int m_nbSides;
    int m_radius;
};

// Derived Class
class Hexagon : public Polygon {
public:
    Hexagon(float radius);
    // 6*radius is easier than generic case
    float computePerimeter();
};
```



Exercise - polymorphism

1. Edit trypoly.cpp, create a Pentagon and an Hexagon and call computePerimeter.
2. Compile and run trypoly.