```
#include <iostream> //reading & writing from keyboard
#include <cmath> //the square root function & absolute value
#include <string>//when use string, include this
#include <vector>//when use vector, include this
#include “.h” //the class head file
#include <fstream> //read and write file
```

```
int main(int argc, char const *argv[])
{
    /* code */
    return 0;
}
```

```
if (/* condition */)
 {
    /* code */
 }
else
    /*code*/
```

```
for (int i = 0; i < count; i++)
 {
    /* code */
```

cout, cin, endl, vector, sort 前面要有 std::

定义的 function 前面要有返回的值的类型。bool return T/F, void 没有 return 的值（可以只打一个 return）

char 型--------只有一个字符

string 型------字符串

```
cout<<”h”<<endl;
cin>>a;
```

std::vector<int> v //建立 vector，<>中为 vector 所存元素类型

v.push_back(s); //想用 vector 先#include

vector 或 string 有.size( )

```
//File: date.h
#ifndef __name_h_
#define __name_h_
class Date{
public:
   Date();
   Date(int aMonth, int aDay, int aYear);
//Accessors
   int getDay() const;
//Modifiers
   void setDay(int aDay);
//other member functions that operate on date class
   bool isEqual (const Date& date2) const;
   void print() const;
private: //representation
   int day;
   int month;
   int year;
};
#endif
```

```
//File: date.cpp
#include <iostream>
#include ”date.h”
Date::Date(){//default constructor
   day=1;
   month=1;
   year=1900;
}
Date::Date(int aMonth, int aDay, int aYear) {
// construct from month, day, & year
   month = aMonth;
   day = aDay;
   year = aYear;
}
int Date::getDay() const { return day; }
void Date::setDay(int d) { day = d; }
bool Date::isEqual(const Date& date2) const
{
   return day == date2.day && month == date2.month && year
== date2.year;
}
void Date::print() const {
   std::cout << month << "/" << day << "/" << year;
}
```

```
Std::sort(v.begin(),v.end(),earlier_date);
```

```
bool earlier_date (const Date& a, const Date& b) {
  if (a.getYear() < b.getYear() ||
      (a.getYear() == b.getYear() && a.getMonth() < b.getMonth()) ||
      (a.getYear() == b.getYear() && a.getMonth() == b.getMonth() && a.getDay() < b.getDay()))
    return true;
  else
    return false;
}
```

Non-member function

.h file

```
// operator< to allow sorting
bool operator< (const Name& left, const Name& right);

// operator<< to allow output
std::ostream& operator<< (std::ostream& ostr, const Name& n);
```

.cpp file

```
// operator<
bool operator< (const Name& left, const Name& right) {
  return left.last()<right.last() ||
    (left.last()==right.last() && left.first()<right.first());
}
// The output stream operator takes two arguments: the stream (e.g., cout) and the object
// to print.  It returns a reference to the output stream to allow a chain of output.
std::ostream& operator<< (std::ostream& ostr, const Name& n) {
  ostr << n.first() << " " << n.last();
  return ostr;
}
```

读取文件

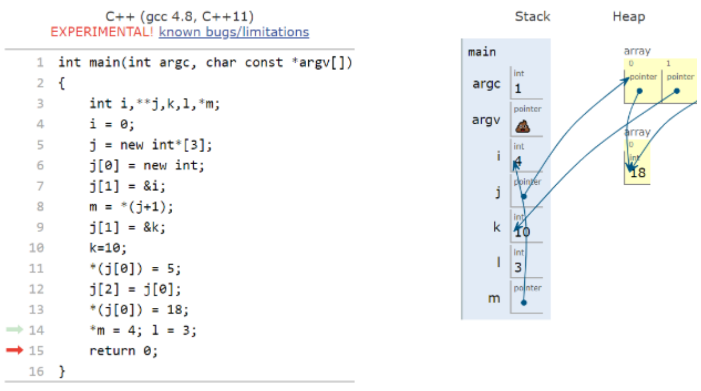
std::ifstream in_str(argv[3]);（读取）

```
while (in_str >> my_variable) {
     // do something with my_variable
   }
```

```
if (!in_str.good()) {
  std::cerr << "Can't open " << argv[3] << " to read.\n";
  exit(1);
}
```

& : 取地址运算符

* : 指针运算符（间接访问运算符）



```
int main(int argc, char const *argv[])
{
    int i,**j,k,l,*m;
    i = 0;
    j = new int*[3];
    j[0] = new int;
    j[1] = &i;
    m = *(j+1);
    j[1] = &k;
    k=10;
    *(j[0]) = 5;
    j[2] = j[0];
    *(j[0]) = 18;
    *m = 4; l = 3;
    return 0;
}
```

空指针：NULL

Passing by reference is more efficient than passing than value.

float a[5]={0,1,2,3,4}

float* p;

a[2] ↔*(a+2)   a[0] ↔ *a

p=a ↔ p=&a[0]   p=a+2 ↔ p=&a[2]

and &&   or ||

#include <string> 字符串切片

Sting.substr(a,b) 从 a 开始切 b 个

字符串转数字

#include<sstream>

Int a;

std::istringstream ia(string);

ia>>a

数字转字符串

#include <string>

Std::to_string(number)

输出靠左：

out_str.width(length_code);out_str<<std::left<<dept1[a].getcode()<<" ";

输出靠右：

std::setw(2) << i*5

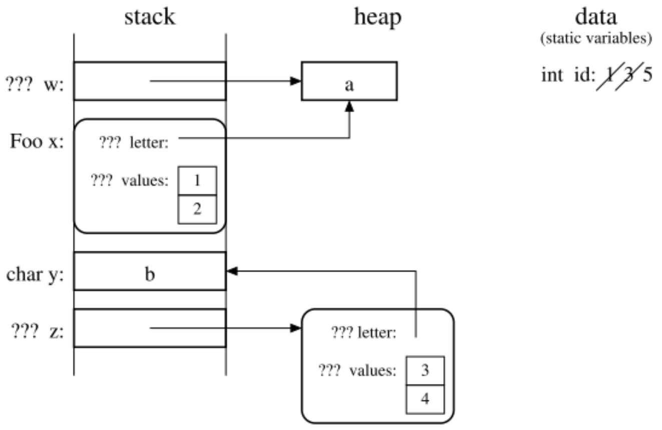
如果要在 function 里面更改导入的参数的值，要用引用传递。

Clean up memory:

delete a[0];

delete [] a[1];

delete [] a;



```
class Foo {
public:
  Foo(char* l);
private:
  char* letter;
  int values[2];
};

Foo::Foo(char *l) {
  static int id = 1;
  letter = l;
  values[0] = id;
  values[1] = id+1;
  id += 2;
}

int main() {
  char* w = new char;
  *w = 'a';
  Foo x(w);
  char y = 'b';
  Foo* z = new Foo(&y);
}
```

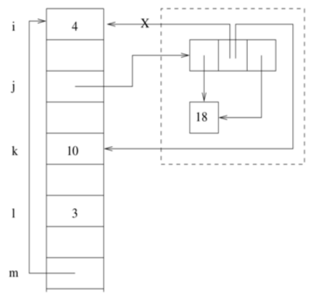
```
#ifndef __student_h_
#define __student_h_
#include <iostream>
#include <string>
#include <vector>
#include "name.h"

class Student {
public:
  // ACCESSORS
  const std::string& first_name() const { return name_.first(); }
  const std::string& last_name() const { return name_.last(); }
  const std::string& id_number() const { return id_number_; }
  double hw_avg() const { return hw_avg_; }
  double test_avg() const { return test_avg_; }
  double final_avg() const { return final_avg_; }
  // MODIFIERS
  bool read(std::istream& in_str, unsigned int num_homeworks, unsigned int num_tests);
  void compute_averages(double hw_weight);
  // PRINT HELPER FUNCTIONS
  std::ostream& output_name(std::ostream& out_str) const;
  std::ostream& output_averages(std::ostream& out_str) const;
private:
  // REPRESENTATION
  Name name_;
  std::string last_name_;
  std::string id_number_;
  std::vector<int> hw_scores_;
  double hw_avg_;
  std::vector<int> test_scores_;
  double test_avg_;
  double final_avg_;
};
```

```
double** a = new double*[rows];
for (int i = 0; i < rows; i++) {
  a[i] = new double[cols];
  for (int j = 0; j < cols; j++) {
    a[i][j] = double(i+1) / double (j+1);
  }
}
```

Consider the following code:

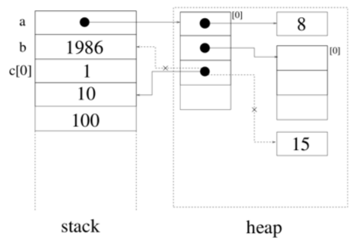
```
int i,**j,k,l,*m;
i = 0;
j = new int*[3];
j[0] = new int;
j[1] = &i;
m = *(j+1);
j[1] = &k;
k=10;
*(j[0]) = 5;
j[2] = j[0];
*(j[0]) = 18;
*m = 4;
l = 3;
```



Consider the following code:

```
int **a;
a = new int*[4];
a[0] = new int;
a[1] = new int[3];
a[2] = new int;
int b = 25;
int c[3] = {1,10,100};

*(a[2]) = 15;
a[0][0] = 8;
a[2] = &b;
a[2][0] = 1986;
a[2] = &(c[1]);
```


a
b
c[0]
1986
1
10
100
[0]
8
[0]
15
stack
heap