

## 教学记录 2

志愿者：郭达雅 时长：3 小时 内容：程序设计期末复习

### Chapter1 Introduction

#### ① programming language:

- known as software,are instrutions to the computer
- programs are written using programming languages

#### Machine language:

- Machine language is a set of primitive instructions built into every computer.
- The instructions are in the form of binary code, so you have to enter binary codes for various instructions.

#### Assembly Language:

- A symbolic representation of the numeric machine codes

#### High—level languages:

- Languages with natural language elements
  - Be easier to use
- need to know what high level languages have.

#### ② Program:

- A sequence of instructions written to perform a specified task with a computer

#### Algorithm:

- an effective method expressed as a finite list of well—defined instructions for calculating a function

#### ③ interpreter:

- translates and excutes a program one statement at a time

### compiler:

—translates the entire source program into a machine—language (binary code) file for execution

### compiler & assembler:

compiler: high—level language to machine language

assembler: assembly language to machine language

#### ④ Programming Errors: syntax errors, runtime error, logic error, common error.

—need to know how to distinguish them

#### ⑤ Know the relation of C++, C, C#, java. And know object—oriented and procedural language

#### ⑥ know how to represent binary、octal、decimal、hex number. octal (0), hex (0X)

往年例题:

1) Which of the following statements about C++ is correct?

- A) C++ is a subset of C.
- B) C++ is developed based on Java.
- C) Any C++ program can be replaced by a C program with equivalent functionality.
- D) C++ is the best programming language.

1) \_\_\_\_\_ translates high-level language program into machine language program. ↵

- A) CPU B) A compiler C) An assembler D) The operating system ↵

2) \_\_\_\_\_ is an object-oriented programming language. ↵

- A) Pascal B) Java C) Ada D) C ↵

1) Which of the following constants is an octal number in C programming? ↵

- A) 843 B) 9x C) 01 D) 0x12 ↵

3) A good programming style can \_\_\_\_\_ ↵

A) make the program more compact ↵

B) prevent reader misunderstandings ↵

C) easier for compiler to find errors ↵

D) make program looks more beautiful ↵

↵

## Chapter 2 Elementary

### ① Identifier (标识符):

- A sequence of characters that consists of
  - letters, digits, and underscores (\_),
- And start with a letter or an underscore (digit? No!),
- Case sensitive.

### ② Reserved word (保留字、关键字)

- A special word that predefined in the language's formal specifications
- It cannot be used as a user-defined name
- Data type names, code construct labels, etc.
  - See Appendix A for a list of reserved words

例题:

2) Which of the following is a valid user-defined identifier?

A) \_x                      B) 9X                      C) x#                      D) else

3) Which of the following is a valid user-defined identifier? ←

A) \_343   B) 9X   C) 8+9   D) class ←

Which of the following identifiers are valid?

miles, Test, a++, --a, 4#R, \$4, #12, apps, main, double, int, x, y, radius

### ③ Variable:

Three elements: name, type and value

A variable must be declared before it can be used.

### ④ how to declare

### ⑥ Constants(常量)

—A value that cannot be changed

Two kinds of constants

—Named Constant (命名常量) etc: `const int size=3;`

—Literal (直接常量) etc: `const int size=3;` 3 is literal

- 4) To improve readability and maintainability, you should declare \_\_\_\_\_ instead of using literal values such as 3.14159. ↵  
A) classes B) constants C) variables D) functions ↵

## ⑦ Numeric Types and Operation

Numeric data types

- Integer number  
int, short, long
  - Modifiers (修饰符)
    - signed vs. unsigned
- Floating-point number  
float, double, long double

注：浮点数不能用作判断，如果用作判断，那么结果往往不确定，除非值相差很大。比如 `4.0-4.0` 不一定等于 `0`

## ⑧ `5/2` & `5.0/2`

`5/2=2`, `5.0/2=2.5`。判断运算的类型，低级向高级转换

## ⑨ increment and Decrement operators

一个表达式是一个值，其中也包含运算。

怎么理解这个概念：

`int a; a=1;` 这里的 `a=1` 是一个表达式，进行操作，将 1 的值赋给 a。  
其中该表达式也是一个值，值为 1。所以 `cout<<a=1<<endl;` 是输出该表达式的值。

`++a`: 表达式的值是 `a+1`，然后 a 的值加 1

`a++`: 表达式的值是 a，然后 a 的值加 1

## ⑩ Type Casting

—implicit casting

```
double d=3;
```

—Explicit casting

```
int i=(int)3.0;
```

注：类型转换不改变原来的值，如：

```
double d=4.5;
```

```
int i=(int) d;
```

最终 i=4,d=4.5 而不是 i=4,d=4

例题：

- 4) Suppose char x='b'; char y = 'a', what is the value of x after "y= x+(x-y)"?   
 A) b B) d C) a D) c
- 5) Suppose x = 0, y = -1, and z = 1. What is the printout of the following statement?   
 if (x > 0 || x == 0)   
 if (y < 0)   
 cout << "x > 0 and y > 0";   
 else if (z > 0)   
 cout << "x < 0 and z > 0";   
 A) x > 0 and y > 0; B) x < 0 and z > 0; C) no printout. D) x < 0 and z < 0;
- 6) Which of the following Boolean expressions is incorrect?   
 A) (4>3)&&(3<=4) B) (x>0)||(x<0) C) -1<x<1 D) 5
- 5) What is the value of (double)(5/2)?   
 A) 2.5; B) 2.0; C) 2; D) 3; E) 3.0;
- 6) Suppose x is a char variable with a value 'b'. What is the printout of the statement cout << ++x?   
 A) b B) d C) a D) c
- 9) Which of the following is a valid/useful boolean expression.   
 A) (1 < x < 100) B) x C) (x <= 5) & (x >= 5) D) (x = 1) || (x != 1)
- 10) What is the value of "1.0 + 1.0 + 1.0 == 3.0"?   
 A) true B) false C) There is no guarantee that 1.0 + 1.0 + 1.0 == 3.0 is true.

- 4) Which of the following math expression is correct?  
A)  $4x+x$       B)  $(x-4)/2(x-5)$       C)  $++(2.0/9+x)$       D)  $x+=(x+1)$
- 5) Suppose `char x = '9'`, what is the printout of `cout << x+1 << endl` ?  
A) 0      B) 58      C) 10      D) A
- 6) Which of the following expression exactly equals to `--i` ?  
A) `i--`      B) `i+=1`      C) `++i`      D) `i=i-1`
- 7) Suppose `x` is a variable of `int`, which of the following is a meaningful boolean expression?  
A) `1 < x < 100`      B) `x = 0`      C) `x`      D) `(x = 1) || (x != 1)`
- 9) What is the value of `"(double)1/3*3 == 1"`?  
A) true      B) false      C) 1      D) maybe true, maybe false

## Chapter 3 Selection

### ① Boolean type

—The data type with two possible values “true” or “false”

Internal representation:

—the value of “true” is 1, another is 0

—any non-zero value is treated as true

bool is=-1;

cout<<is<<end;

the result:1 why? Thinking

### ② conditional operator

y= (x>0) ?1:-1;

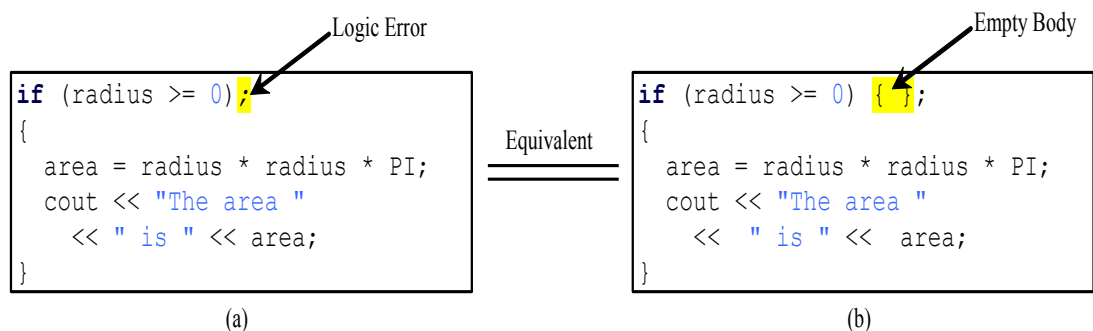
if(x>0),y=1

else y=-1;

### ③ three control structures

- Sequence: 顺序
- Selection (branching): 选择
- Loop (Iteration): 循环

### ④ A common mistake



Not a compiler error;

Not a runtime error;

A logic error !




### ⑤ Dangling “else”

The else clause matches the most recent if clause in the same block

```
int i = 1;
int j = 2;
int k = 3;

if (i > j)
    if (i > k)
        cout << "A";
    else
        cout << "B";
```



### ⑥ short-circuit (短路) operator

&&: conditional or short-circuit AND operator

p1 && p2

- C++ first evaluates p1,
- if p1 is true, then evaluates p2;
- if p1 is false, it does not evaluate p2.

||: conditional or short-circuit OR operator

p1 || p2

- C++ first evaluates p1,
- if p1 is false then evaluates p2;
- if p1 is true, it does not evaluate p2.

注：上面的准则一定要弄懂，非常有用

## ⑦ switch

The case branch is executed when the value in the case statement matches the value of the switch-expression.

- value1, ..., and valueN are **different** constant expressions
- they cannot contain variables in the expression, such as  $1 + x$

- a value of integer
- enclosed in parentheses.

```
switch (switch-expression) {
    case value1: statement(s)1;
        break;
    case value2: statement(s)2;
        break;
    ...
    case valueN: statement(s)N;
        break;
    default: statement(s)-for-default;
}
```

switch—expression 必须整数

value1—valueN 必须不同的常数

当匹配到时，会一直执行下去，除非遇到 break

## ⑧ Operator Precedence

var++, var-- , static_cast()
+, - (plus and minus), ++var, --var
(type) Casting
! (Not)
*, /, %
+, - (addition and subtraction)
<, <=, >, >=
==, !=;
&& (AND)
(OR)
=, +=, -=, *=, /=, %= (Assignment operator)

例题:

┘

7) What is y after the following switch statement is executed? ↵

```
x=1; ↵  
switch(x+=1){ ↵  
    case 1: y =0; break; ↵  
    case 2: y = 1; break; ↵  
    default: y+=1; ↵  
}
```

A) 0    B) 1    C) 2    D) 3 ↵

7) Suppose x = 1, y = -1, and z = 1. What is the printout of the following statement? ↵

↵

```
if (x > 0) ↵  
if (y > 0) ↵  
cout<<"x > 0 and y > 0"; ↵  
else if (z > 0) ↵  
cout<<"x < 0 and z > 0"; ↵
```

A) x > 0 and y > 0;    B) x < 0 and z > 0;    C) no printout.    D) x < 0 and z < 0;    ↵

↵

## Chapter 4 Characters and Strings

### ① char

—the data type for a single character

—special characters

Character	Escape Sequence	Name	ASCII Code
\b		Backspace	8
\t		Tab	9
\n		Linefeed	10
\f		Formfeed	12
\r		Carriage Return	13
\\		Backslash	92
\'		Single Quote	39
\"		Double Quote	34

—Represented by ASCII code

—char & int: int—>ASCII code—>char, 反之亦然

```
int i='a';<==>int i=(int)'a';
```

i 的值是'a'的 ASCII 码

—All numeric operators can be applied to char operands

you just regard char as integer and its value is ASCII code

注：输出时，注意输出类型。

```
char c='a';
```

```
cout<<c<<endl;
```

```
cout<<c+1<<endl;
```

```
c=c+1;
```

```
cout<<c<<endl;
```

result: a

98

b

why?c is char type, while c+1 is int type

## ② String

```
string message = " Welcome to C++ ";  
message += " and programming is fun";  
string s3 = s1 + s2;  
s3 += '!' ;
```

```
string s = "ABC" + "DE";  
string s = message + "ABC" + "DE";
```

字符串 **string** 必须作为左值

比如    `string s="abc";`  
         `s+"dce";`(这样是正确的)  
         `"dce"+s;` (这样是错误的)

上面的例子，`message+ "ABC" + "DE"` 正确，是因为 `message+ "ABC"` 是一个 **string** 类型，然后再加上 `"DE"`。

## ③ simple file input and output

- for infile, you need to make sure whether it exists
- for outfile, you need to make sure the file does not exist or none use the file.If someone use,it will runtime error.

例题：

5) Suppose char x = '9', what is the printout of `cout << x+1<<endl` ?

- A) 0                      B) 58                      C) 10                      D) A

20) In which of the following case can a file be opened to write?

- A) The file does not exist.  
B) The file is being used by another program for input.  
C) The file is being used by another program for output.  
D) The file is being used by another program for both input and output.

6) Suppose x is a char variable with a value 'b'. What is the printout of the statement `cout << ++x?`

- A) b    B) d    C) a    D) c

## Chapter 5 Loop

### ① continue & break

```
for (int i=0;i<n;++i)
{
    loop body;
}
```

对于这循环，其实++i 在 loop body 后面执行，即

```
for (int i=0;i<n;)
{
    loop body;
    ++i;
}
```

如果是 continue，则会直接跳到 loop body 后面，直接执行++i，然后继续循环。

如果是 break，则会直接跳出循环，不会执行++i。

### ② “do—while” loop & “while” loop

— “do—while” loop will execute loop body firstly

— “while” loop will judge the condition firstly

例题：

8) What is sum after the following loop terminates? ↵

```
int sum = 0, item = 0; ↵
do{ ↵
    item++; ↵
    sum += item; ↵
    if(sum>4) break; ↵
} while(item<5); ↵
```

A) 5      B) 6      C) 7      D) 8 ↵

# Chapter 6 Function

## ① Basics of Function

- Definition of Function

—A collection of statements that are grouped together to perform an operation.

- Benefits of Function

—Code reusing

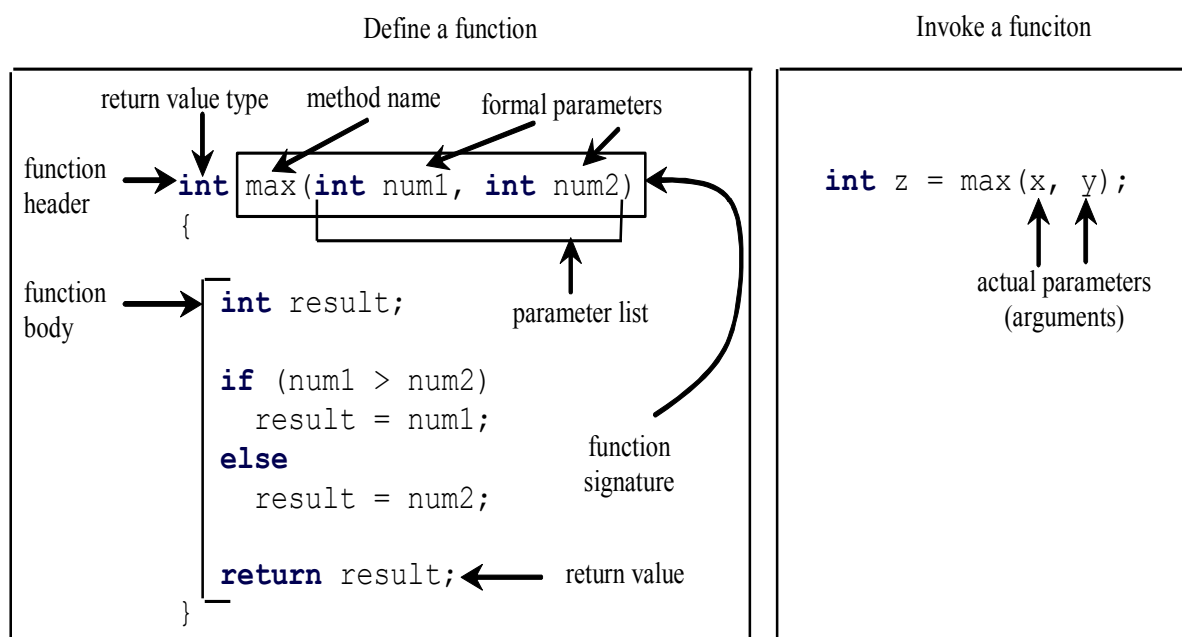
Avoid re-coding, re-compiling

—Code Modularizing

“Divide and conquer”

—Information hiding

Hide the implementation from the user



## ② Function Header

- Function header

— Return value type + function name + parameter list

- Function signature (签名)

— Function name + parameter list

- Formal parameter (形式参数)
  - The variables defined in the function header
  - Parameters are optional
- Return Value Type (返回值类型)
  - The data type of the value the function returns
  - Void function:
    - no return value and return value type is “void”

### ③ Function Body (函数体)

- “return” statement is required for value-returning functions
  - A function terminates when a return statement is executed
- Although there may be some statements left

### ④ Parameter

#### Two method of passing parameters

##### —Pass by value (值传递)

只是传递值，即拷贝一份值，函数中修改该值，不会影响原来的值

##### —Pass by reference (引用传递)

直接传递变量，不需要拷贝，所以速度更快

- Pass-by-Value
  - Single direction transfer
    - Simple and safe
  - Need “copying” values
- Pass-by-Reference
  - Two direction transfer
  - No need copying values
    - More efficient for complex data types, e.g. string

### ⑤ Overloading

- two functions within one file with
  - The same name, but
  - Different parameter lists



- The types and/or the number of arguments
- return value type does not matter

## ⑥ Ambiguous invocation

- No exactly matching version
- More than one matching after conversion

Will cause a compilation error!

## ⑦ Default Arguments (缺省参数)

Note: Default arguments must be declared at last

<code>int max(int x,int y=0,int z=0){}</code>	right
<code>int max(int x,int y=0,int z){}</code>	wrong

## ⑧ Function prototype (原型)

- A function declaration without implementation (the function body).
- The implementation can be given later in the program, after the calling of the function

- Function prototypes are usually put in the beginning of a program
  - To make sure the callings occur after declaration

## ⑨ inline function

## ⑩ Variables

Local variable & global variable

Local variable

- A variable defined inside a function.
- The scope of a local variable
  - From its declaration and continues to the end of the block that contains the variable.

Note: Initialize a local variable before using it!

- **The lifetime of a local variable**
  - Part of the lifetime of the entire program
    - The execution time of the variable's code block (its scope)

## Global Variable

- Declared outside any function
- **The scope of a global variable**
  - With global scope
    - Accessible to all functions

**Note:** A global variable is automatically initialized by the system to zero

- **The lifetime of a global variable**
  - The lifetime of the entire program

## ⑪ Static local variables

- Global lifetime
  - Permanently allocated in the memory for the lifetime of the program
- Local Scope
  - The same scope as a non-static variable
- Initialization
  - Only once!!!
  - By the system( to zero) or by user

# Check Points

Identify the variable types in the following program

```
int x = 0;
void show(int k) {
    static int j=0;
    j++;
    cout<<k<<endl<<j<<endl;
}
int main() {
    int i;
    for(i=0; i<3; i++){
        cin>>x;
        show(x);
    }
}
```

当 x=1, 3, 5 时。搞清楚输出是什么。

例题:

- 10) Which of the following statements about variable is not correct?
- A) A local variable will be automatically initialized to zero if no initial value is given.
  - B) The address of a variable is the address of the first byte of the variable's memory space.
  - C) The type of a variable determines the size of memory space used by the variable.
  - D) It is possible that two variables have the same name.
- 11) Which of the following is not an advantage of using functions?
- A) To hide detailed implementation from the client/caller.
  - B) To ease code reusing.
  - C) To make programs easier to read.
  - D) To make programs run faster.

- 12) Which of the following statements about function is correct?
- A) A function can be defined inside another function.
  - B) The implementation of a function may appear after the function is called.
  - C) A value can only be passed from a callee to its caller using a return statement.
  - D) A default parameter must be the last parameter in any parameter list.
- 13) To overload a function, you must define functions with different:
- A) headers      B) names      C) return types      D) signatures
- 14) Which of the following about function matching is not correct?
- A) A function cannot be matched if it has a different name.
  - B) A function cannot be matched if it has a different signature.
  - C) A function cannot be matched if it has no return value.
  - D) A function cannot be matched if it has not enough number of arguments.
- 18) Which of the following statement is not correct?
- A) The scope of a local variable is within the function it is defined.
  - B) The scope of a static variable is within the function it is defined.
  - C) The lifetime of a local variable is equal to that of the program.
  - D) The lifetime of a static variable is equal to that of the program.
- 10) Which of the following is NOT an advantage of using functions? ↵
- A) Enabling the divide and conquer programming methodology ↵
  - B) Providing logic abstraction ↵
  - C) Reusing similar program logic ↵
  - D) Hiding data from logic ↵
- 11) To overload a function, you must declare function versions different in \_\_\_\_\_. ↵
- A) parameter list      B) function name      C) return type      D) function header ↵
- 12) Which of the following statement is true? ↵
- A) You can return a variable with any type ↵
  - B) To call a function, the number of arguments must be the same as the number of parameters ↵
  - C) A function may not return any value to the caller ↵
  - D) The functions must be declared in the same order as they are called ↵

## Chapter 7 Array

### ① Array

—Array is a data structure that represents a collection of the same type of data.

**Note: the array size in the declaration must be a constant expression.**

```
int size = 4;    double myList[size];           wrong
const int size = 4; double myList[size];       right
```

The bound of an array

- The scope of the index: 0..arraysize-1 (小心越界)
- C++ compiler does not check array's bound!!!

### ② Initializing Arrays

Default initialization by system

- To arbitrary values on creation

#### • Implicit Size

- To omit the array size when declaring and creating an array using an initializer
- C++ automatically figures out how many elements are in the array

Example:

```
double myList[] = {1.9, 2.9, 3.4, 3.5};
```

#### • Partial Initialization

- To initialize a part of the array
- The rest elements are initialized to “0” or “a”

Example:

```
double myList[4] = {1.9, 2.9};
```

```
double myList[4] = {};
```

**Note: initializing must be with declared.**

### ③ Array as Parameter

—Array is passed by reference

—The starting address is passed to the function

#### ④ Multi-D Array

- You can pass a two-dimensional array to a function
  - The column size to be specified in the function declaration.

```
int sum(const int a[][COLUMN_SIZE], int rowSize)
```

例题:

14) Which of the following statements can execute successfully? ↵

A) `double x[5]; cout<<x[5];` ↵

B) `int x; cin>>x; int b[x];` ↵

C) `char ch[10]; cin>>ch;` ↵

D) `double d[3]; d = {1, 2, 3};` ↵

15) Which of the following function declarations is correct? ↵

A) `int f(int [][] a, int size1, int size2);` ↵

B) `int f(int a[][], int size1, int size2);` ↵

C) `int f(int a[][5], int size1);` ↵

D) `int f(int a[4][5]);` ↵

↵

17) How to initialize an array of 10 double 0's ?

A) `double array[] = {0};`    B) `double array[10] = {0};`

C) `double array[] = 0;`    D) `double array[10] = 0;`

## Chapter 11 Pointer

### ① Pointer

—the variable to hold memory addresses as its value

### ② Initialize a Pointer

- Implicit initialization
  - A local pointer is assigned an arbitrary value
  - A global pointer is assigned to NULL (pointing nothing)

NULL is in fact “00000000”

- Explicit initialization

```
int count=5;
int *pCount = &count;
int *ptr = NULL;
```

Note: The type of the address assigned to a pointer must be consistent with the pointer type.

For example,

```
int area = 1;
double *pArea = &area;    //wrong
```

### ③ Indirect Reference

*Indirection*: referencing a value through a pointer.

For example:

```
int count=5;
int *pCount = &count;
count++; //direct reference
(*pCount)++; //indirect reference
```

### ④ p vs \*p vs &a

—need to figure out them

## ⑤ Some special pointers

### Constant Pointer (指针常量)

—A pointer whose value can not be changed

```
double radius = 5;
double * const pValue = &radius;
(*pValue) = 3.0; //right
pValue ++;        //wrong
```

### Pointer of Constant (常量指针)

—A pointer that points to a constant

```
double radius = 5;
const double * pValue = &radius;
(*pValue) = 3.0; //wrong
radius = 3.0;    //right
```

## ⑥ Pointer and Array

—Array variable is different from a pointer

Array name is a constant (not variable) address

## ⑦ Pointer in Function

- Pointer as parameter

—The address stored in the pointer is passed to the function

—A kind of pass-by-reference

Note: for pointer, it's pass by value.but for the object, it's pass by reference

## ⑧ New and delete

new:

```
int *pValue = new int;
int *mylist = new int[10];
```

delete:

```
delete pValue;
delete [] mylist;
```



## ⑨ C—string

### ○ Two ways to process strings

- To treat strings as arrays of characters
  - known as *pointer-based strings* or *C-strings*

'D'	'a'	'l'	'l'	'a'	's'	'\0'
-----	-----	-----	-----	-----	-----	------

- To process strings using the string class
  - The string class will be introduced in Chapter 9

### ○ Two ways to declare C-string

- Using an array variable

```
char city[7] = "Dallas";
```

```
char city[] = {'D', 'a', 'l', 'l', 'a', 's', '\0'};
```

```
char city[] = "Dallas";
```

```
cout << city;
```

```
char city[7]; city= "Dallas";
```

'D'	'a'	'l'	'l'	'a'	's'	'\0'
city[0]	city[1]	city[2]	city[3]	city[4]	city[5]	city[6]

```
char city[10];  
cin >> city;
```

```
char *city2;  
cin >> city2;
```

No memory is allocated to store the content of city2.

例题:

13) Given: `int i, int j=2; int *p=&i;` which of the following statement can perform the same assignment as "`i=j;`"? ↵

A) `i = *p;` ↵

B) `*p = *&j;` ↵

C) `i = &j;` ↵

D) `i = **p;` ↵

↵

16) Assume you declared `int *p` and `p`'s current value is 1000. What is `p + 1`? ↵

A) 1002

B) 1003

C) 1004

D) 1001 ↵

16) To declare a pointer, you can use \_\_\_\_\_

- A) `int *p;`    B) `int p*;`    C) `int &p;`    D) `int (*P);`

17) Suppose `int count = 5`, what is the value of `&count`?

- A) the reference of count;    B) 5;    C) the address of count;    D) the pointer of count;

18) What is the printout of the following statements?

```
char *p = "abcd";
```

```
p+=2;
```

```
cout<<p;
```

- A) `abcd`    B) `ab`    C) `cd`    D) memory error

19) Suppose `int *list = new int[5]`, how to delete the array?

- A) `delete *list;`    B) `delete list[5]`    C) `delete list[]`    D) `delete [] list`

14) Suppose you declare

```
int count = 5;
```

```
int *pCount = &count;
```

which of the following is true?

- A) `*pCount` is 5    B) `"count"` contains the address of count  
C) `*count` is the address of count    D) `&count` is 5

15) Suppose you declare the following:

```
double radius = 5;
```

```
const double const* pValue = &radius;
```

15) Suppose you declare variables as below, which of the following statements is true?

```
int i = 10;
```

```
int *pi = &i;
```

```
int &ri = i;
```

- A) `*pi` is 10    B) `&i` is 10  
C) `pi` contains the value of `i`    D) `i` contains the address of `i`

16) Which of the following statements about array is correct?

- A) An array is exactly a pointer to a sequence of variables.  
B) The computer will automatically check if each array index is out of bounds in run-time.  
C) When an array is used as an argument, only the address of the array will be passed.  
D) When an array is used as an argument, its length must be passed as another argument.

## Chapter 17 Recursion

### ① Recursion

- High cost in both time and memory
- Any recursive function can be converted to non-recursive (iterative) function
- easy to solve “recursive” problems

### ② how to use?

- see my ppt “排序和递归”

例题:

19) Which of the following statements about recursion is correct?

- A) A recursive function must call itself in its function body.
- B) An iterative function is easier to be designed than a recursive function with the same functionality.
- C) Recursion runs slower, so it should not be used if the problem can be solved by iteration.
- D) Any recursive function can be replaced by an iterative one with the same functionality.

20) Which of the following statements is NOT true?↵

- A) Every recursive function must have a base case or a stopping condition
- B) Every recursive function can be converted to an iterative one↵
- C) Every recursive function must have a return value↵
- D) Recursion is not always helpful↵