Chapter 4~6

① 以下两个程序段不等价，执行程序段B将陷入死循环。

```
/* 程序段A*/
s = 0;
for(i = 1; i <= 10; i++) {
    if(i % 2 == 0){
        continue;
    }
    s = s + i;
}
```

```
/* 程序段B */
s = 0;
i = 1;
while(i <= 10){
    if(i % 2 == 0){
        continue;
    }
    s = s + i;
    i++;
}
```

→ for 语句的 continue 会执行 i++ ；

while 语句则不会额外执行

（for 和 while 等价前提：无 continue）

V

○ T ◉ F

答案错误：0 分

② 写出以下2个程序段的运行结果。

```
/* 程序段A */
int num = 0;
while(num <= 6){
    num++;
    if(num%3 == 0){
       continue;
    }
    printf("%d#",num);
}
```

num=7 也会被下述程序判断 → 注意条件判断和 num++ 的顺序

→ for (__; ④; __)

会在最开始执行一遍，故也考虑顺序问题

程序段A的输出结果是 1#2#4#5# 1 分

③ 以下程序段（）的功能是：计算1! + 2! + 3! + … + n!。假设变量和函数都已正确定义。

A.
```
sum = 0;
for(i = 1; i <= n; i++ ){
     sum = sum + fact (i);     /*  假设函数fact(i)已正确定义 */
 }
 printf("%.0f\n", sum);
```

B.
```
sum = 0;
for(i = 1; i <= n; i++){
     item = 1;
     for (j = 1; j <= i; j++)
         item = item * j;
     sum = sum + item;
}
printf("%.0f\n", sum)
```

C.
```
sum = 0;
item = 1;
for(i = 1; i <= n; i++){
     for (j = 1; j <= i; j++)
         item = item * j;
     sum = sum + item;
}
printf("%.0f\n", sum)
```

→ 注意嵌套循环中的重置变量

D.
```
for(i = 1; i <= n; i++){
     sum = 0;
     item = 1;
     for (j = 1; j <= i; j++)
         item = item * j;
     sum = sum + item;
}
printf("%.0f\n", sum);
```

E.
```
sum = 0;
item = 1;
for(i = 1; i <= n; i++){
    item = item * i;
    sum = sum + item;
}
printf("%.0f\n", sum);
```

④ 若变量已正确定义，写出执行以下程序段后变量的值。请注意，直接填数字，前后不要加空格等任何其他字符。

```
scanf ("%d", &m);
is_prime = 1;
limit = sqrt(m) + 1;
for(i = 3; i <= limit; i += 2){
    if(m % i == 0){
        break;
        is_prime = 0;
    }
}
```

→ 注意 break 和变量改变语句的顺序

输入 11，变量 is_prime 的值是 1 1 分

输入 25，变量 is_prime 的值是 0 1 分

⑤ 以下正确的函数定义形式是（）。

A. double fun(int x, int y)

B. double fun(int x ; int y)

C. double fun(int x, int y); → 函数声明

D. double fun(int x, y)

答案错误：0 分

⑥ 输入一个非负整数，从高位开始逐位分割并输出它的各位数字。例如，输入9837，输出9 8 3 7

```
int digit, number, pow, t_number;
scanf ("%d", &number);
t_number = number;
pow = 1;
while [ t_number >= 1 ] 1 分 {
    pow = [ pow * 10 ] 1 分 ;
    t_number = t_number / 10;
}
while ( pow >= 1 ) {
    digit = [ number / pow ] 1 分 ;
    number = [ number % pow ] 1 分 ;
    pow = [ pow / 10 ] 1 分 ;
    printf ("%d ", digit);
}
printf ("\n");
```

已经作为变量，应用于实现对应函数的功能，而不是调用同名函数

⑦ static int f = 1;

⑧ if (a == 1) { .. }

⑨

```
char ch;
while((ch = getchar()) !='#'){
    putchar(ch);
    ch = getchar();
}
```

2次 getchar() 易忽略

输入 123456#，输出 [123456] 1 分

⑩ case 和 default 等 加 break

⑪ %f 保留6位注意四舍五入

⑫ 表达式 (z=0, (x=2)||(z=1),z) 的值是1。

T ✓F

已经为1，(z=1) 不执行 短路现象☆

⑬ 一维数组定义的一般形式如下，其中的类型名指定数组变量的类型。

```
类型名 数组名[数组长度];
```

数组中每一个元素的类型 整体

T F

答案错误：0 分

⑭ 有符号的短整型（16bits）二进制表示 1111111111101100，对应十进制数字 -20

原码 0000000000010100
反码 1111111111101011
用补码表示
原码取反+1

⑮ 假设 int x = 11; 表达式 x++*1/3 的值为( )。

x++中++在后，先算式子求值再 x=x+1.

A. 0

✓B. 3

C. 3.666667

D. 4

答案错误：0 分

⑯ 若a是32位int类型变量，判断其32个2进位中末两位均为1的表达式为（ ）。

✓A. (a&3)==3 位运算仍以十进制表示

B. (a&3)==11

C. (a&11)==3

D. (a&11)==11

答案错误：0 分

阶段性测试 2

① 以下程序段的功能是输出1～100之间每个整数的各位数字之和。

```
for (num = 1; num <= 100; num++) {
    s = 0;
    do {
```

内外层循环都使用num

② * 改成 unsigned int i 只能表示有限大小，到max+1后变回0

■ 运行包含以下代码段的程序将可能进入死循环。

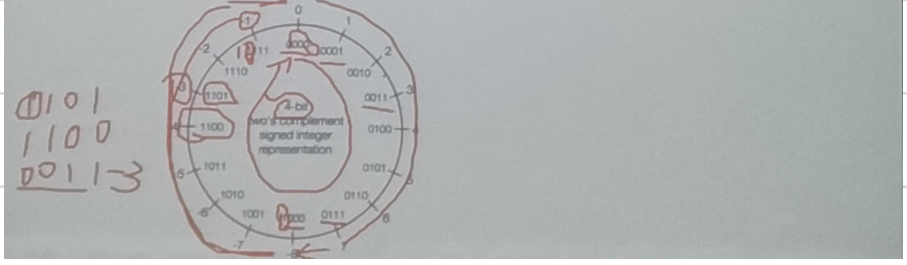
```
int i =-1;
while (i > 0) {i++; printf("%d ", i); }
```

□ 考察整数溢出：-2147483648~2147483647 -2^{31}~$2^{31}-1$

```
    s = s + num % 10;
    num = num / 10;
  } while (num != 0);
  printf("%d\n", s);
}
```

do { } while(...);循环内是一个语句，可省略 {}，语句结尾有分号

内层与外层循环使用不同变量，每次内层循环后，num为0



③

- 已知字符B的ASCII码是66，那么也可以通过转义符\，用'\66'来表示字符常量B。 ✗

十进制不是八进制

转义字符表

字符形式	所表示字符
\n	换行
\t	横向跳格（即输出若干个空格）
\b	退格（显示输出时，刷新左边一个字符）
\r	回车（输出位置重新移到行首）
\\	反斜杠字符"\"
\'	单引号（撇号）
\ddd	八进制数ddd所代表字符，如\007为"嘟"声，\40即空格。
\xhh	十六进制数hh所代表的字符，如\x41即'A'，\x20即空格。

④

- 若变量已正确定义并赋值，符合C语言语法的表达式是__.

A. a=2++

□ ++ --只能作用在变量上，a++相当于a = a + 1，返回原a

B. a=3,5 ✓

□ 逗号表达式，(a=3), 5, 表达式值为5

C. a=a+1=3

□ =右结合，a=(a+1)=3)，=左侧只能是变量(内存存储单元)

D. 12.3%4

□ %只能作用在整数上

⑤

- 以下（ ）不是正确的函数原型？

A. int f();

B. int f(int i);

C. int f(int); ✓

D. int f() {}

可省略形参名

函数定义不能省{}内要调用

- 只写函数定义中的第一行(函数首部)，并以分号结束

函数类型 函数名(参数表);

⑥

- 已知字母A的ASCII码为十进制数65，且ch为字符型变量，则执行语句ch='A'+'6'-'3';后，ch中的值为__。(往年试卷)

A. 'D'

B. 68 ✓

C. 'C'

D. 不确定

⑦ sizeof 不执行内部语句 只推导结果类型

⑧

- 以下正确的字符常量是__B__。

A. '\412'

B. 255

C. '%d'

D. '\'

8 0~255 8 bytes，仅到255

4×64+1×8+2×1=255

''内有两个字符

转义符表示一个字符

缺一个字符

⑨ 结论：~a (a≥0) = -(a+1)

⑩

```
# include <stdio.h>
int s;
int f(int m){
   static int k=0;
   for(; k<=m; k++) s++;
   return s;
}
void main(void) {
   int s=1;
   s=f(2)+f(2);
   printf("%d#%d#", s, f(20));
   return;
}
```

代码	全局s	main.s	f.m	f.k
	0			0
int s=1	0	1		0
第一次f(2)	0	1	2	0
k=0	0	1	2	0
for	3	1	2	3
f(2)返回3				
第二次f(2)	3	1	2	3
for	3	1	2	3
f(2)返回3				
s=f(2)+f(2)	3	6		3
f(20)	3	6	20	3
for	21	6	20	21
f(20)返回21				

⑪

阶段性测试2

```
#include <stdio.h>
void swap(int x, int y) {
   x = x ^ y;
   y = x ^ y;
   x = x ^ y;
}
int main(void) {
   int x = 5, y = 2;
   swap(x, y);
   printf("%d#%d", x, y);
   return 0;
}
```

内存

0xFFFFFFFF

Stack

main: x 5, y 2

swap: x 2, y 5

实参 → 形参 单向传递

常见逻辑错误：将实参和形参看成同一变量

0x00000000

⑫

阶段性测试2

```
#include <stdio.h>
int main() {
   int i, j, k=0;
   for (i = 0; i < 10; i++) {
      for (j = i; j < 10; j++) {
         if (j%2 == 0) { continue; }
         if (j%3 == 0) { break; }
         k+=j;
         j++;
      }
   }
   printf("%d", k);
   return 0;
}
```

i	j	k
0	0 (continue)	0
0	1	0+1=1
0	3 (break)	
1	1	1+1=2
1	3 (break)	
2	2 (continue)	
2	3 (break)	
3	3 (break)	
4	4 (continue)	
4	5	2+5=7
4	7	7+7=14
4	9 (break)	
5	5	14+5=19
5	7	19+7=26
6	6 (continue)	
6	7	26+7=33
6	9 (break)	

⑬

第八章 指针

1、以下选项中，对基本类型相同的指针变量不能进行运算的运算符是（）。

A.+

B.-

C.=

D.==

答案：A

解析：“<（小于）”运算在两个同类型的指针间可以比较大小，比较原则应该是按照实际内存的高低位比较的；

“=（等于）”是对于类型相同的两个指针变量之间常规运算；

“-（减法）”运算两个相同指针变量相减可以获得在之间相隔的同类型元素个数（在某个类型的数组中的应用）；

“+”运算是不可以的，因为两个指针相加什么都得不到，所以规定不允许相加。

14

下述对C语言字符数组的描述中错误的是（）。

A. 字符数组可以存放字符串

B. 字符数组中的字符串可以整体输入、输出

C. 可以在赋值语句中通过赋值运算符"="对字符数组整体赋值

D. 不可以用关系运算符对字符数组中的字符串进行比较

常三次小测前复习

段错误-写不属于自己/非法内存单元

传值作为地址

```
int n;
scanf("%d", n);
```

n未初始化，是不确定值(如39)，scanf将数据写入到不确定值(如39)地址，而非n的地址，即使没有导致段错误，n值未改变

数组越界访问

```
int i = 0, j = 0, a[10];
for (i = 0; j < 10; i++)
    a[i] = i;
```

*嵌套循环记得使用不同变量

数组越界访问

```
int i = 0;
char s[100];
while ((s[i] = getchar()) != '\0')
    i++;
```

键盘无法输入字符'\0'，导致数组越界访问

指针未初始化/未指向有效内存单元

```
int *p, *q = NULL;
scanf("%d", p);     *p = 5;
scanf("%d", q);     *q = 5;
```

NULL为特殊内存地址，写入该地址导致段错误。建议所有指针都初始化为NULL，暴露未指向有效内存单元的指针

修改常量区的数据

```
char *str = "Hello";
scanf("%s", str);
str[0] = 'h';
```

字符串"Hello"存储在程序内存的常量区，指针变量str指向常量区，常量区的内存单元不能修改

107

死循环 → 数组/指针越界访问

```
while (*s != '\0') {
    *(t+i) = *(s+i);
    i++;
}
```

死循环导致字符数组/指针t越界访问

```
while (*(s+i) != '\0') {
    *(t+i) = *(s+i);
    i++;
}
printf("%s", t);
```

printf直到遇到字符串结束符'\0'结束

Strings In Memory

- 1. If we create a string as a char[ ], we can modify its characters because its memory lives in our stack space.
- 2. We cannot set a char[ ] equal to another value, because it is not a pointer; it refers to the block of memory reserved for the original array.
- 3. If we pass a char[ ] as a parameter, set something equal to it, or perform arithmetic with it, it's automatically converted to a char *.
- 4. If we create a new string with new characters as a char *, we cannot modify its characters because its memory lives in the data segment.
- 5. We can set a char * equal to another value, because it is a reassign-able pointer.
- 6. Adding an offset to a C string gives us a substring that many places past the first character.
- 7. If we change characters in a string parameter, these changes will persist outside of the function.

可修改元素值

不能用等号整体赋值

STRCPY可

数组初始化后长度不会变

函数传递时会退化成指针

不可修改元素值

会段错误 ⇔ const int a = 1; a = 2; 编译错误

可整体操作

可添加偏移量得到字符串

1

已有定义 `int k=2; int *ptr1, *ptr2;` 且 `ptr1` 和 `ptr2` 均已指向变量 `k`，下面不能正确执行的赋值语句是( )。

A. k = *ptr1 + *ptr2;

B. ptr2 = k;

C. ptr1 = ptr2;

C. ptr1 = ptr2;

D. k = *ptr1 *(*ptr2);

②

下面程序段的运行结果是（）。

```
char s[ ] = "language", *p = s;

while( *p++ != 'u') {
    printf("%c", *p - 'a' + 'A');
}
```

勿忽略条件判断语句中变量值的改变

A. LANGUAGE

B. ANGU

C. LANGU

D. LANG

③

在64位程序中，`sizeof("\num\\\t")`的值为 6 1 分，`strlen("\num\\\t")`返回 5 1 分，`sizeof("\num\\\t" + 1)`的值为 8 1 分，`strlen("\num\\\t" + 1)`返回 4 1 分。

sizeof 退化成指针 64位 int ⇒ 8

地址+1

④

根据下面的定义，能打印出字母`M`的语句是（）。

```
struct person{
  char name[10];
  int age;
} c[10] = { "John", 17, "Paul", 19, "Mary", 18, "Adam", 16 };
```

A. printf("%c", c[3].name);

B. printf("%c", c[3].name[1]);

C. printf("%c", c[2].name[0]);

D. printf("%c", c[2].name[1]);

⑤

以下程序的输出结果是（）。

```
struct stu{
   int x;
   int *y;
} *p;
int dt[4] = {10, 20, 30, 40};
struct stu a[4] = {50, &dt[0], 60, &dt[1], 70, &dt[2], 80, &dt[3]};

int main( )
{
   p=a;
   printf("%d,", ++p->x);
   printf("%d,", (++p)->x);
   printf("%d", ++(*p->y));

   return 0;
}
```

A. 10,20,20

B. 50,60,21

C. 51,60,21

D. 60,70,31

⑥

```
#include <stdio.h>
struct st{
```

```
  char c;
  char s[80];
};
struct  st a[4] = {{'1',"123"}, {'2',"321"}, {'3',"123"}, {'4',"321"}};
char * f(struct st *t);

int main( )
{  int k;

  for(k = 0; k < 4; k++){
    printf("%s", f(a+k));
  }

  return 0;
}

char * f(struct st *t)
{
   int k = 0;

   while(t->s[k] != '\0'){
        if( t->s[k] == t->c){
          return t->s+k;
        }
        k++;
    }

   return t->s;
}
```

不是 char，是 substring

7. 运行以下程序，第一个 `printf` 输出 1#1#0 1 分，第二个 `printf` 输出 1#1#6 1 分，第三个 `printf` 输出 1#1#6 1 分，第四个 `printf` 输出 3#6#9 1 分。

```
#include <stdio.h>
struct st {
  int a;
  int b[5];
};

void f1(struct st v, int b[])
{
    v.a = 2;
    v.b[0] = 4;
    b[1] = 6;
}

void f2(struct st *v, int *b)
{
    v->a = 3;
    (*v).b[0] = 6;
    *(b+1) = 9;
}

int main()
{
    struct st s1 = { 1, {1} };
    printf("%d#%d#%d", s1.a, s1.b[0], s1.b[1]);

    f1(s1, s1.b);
    printf("%d#%d#%d", s1.a, s1.b[0], s1.b[1]);

    struct st s2 = { 2, {2, 2} };
    s2 = s1;
    printf("%d#%d#%d", s2.a, s2.b[0], s2.b[1]);

    f2(&s2, s2.b);
    printf("%d#%d#%d", s2.a, s2.b[0], s2.b[1]);

    return 0;
```

等价（退化）

8.

https://www.cnblogs.com/tanrong/p/7405919.html

printf 的顺序问题

他是采用栈执行的

所以是先进后出

最后一个函数最先执行

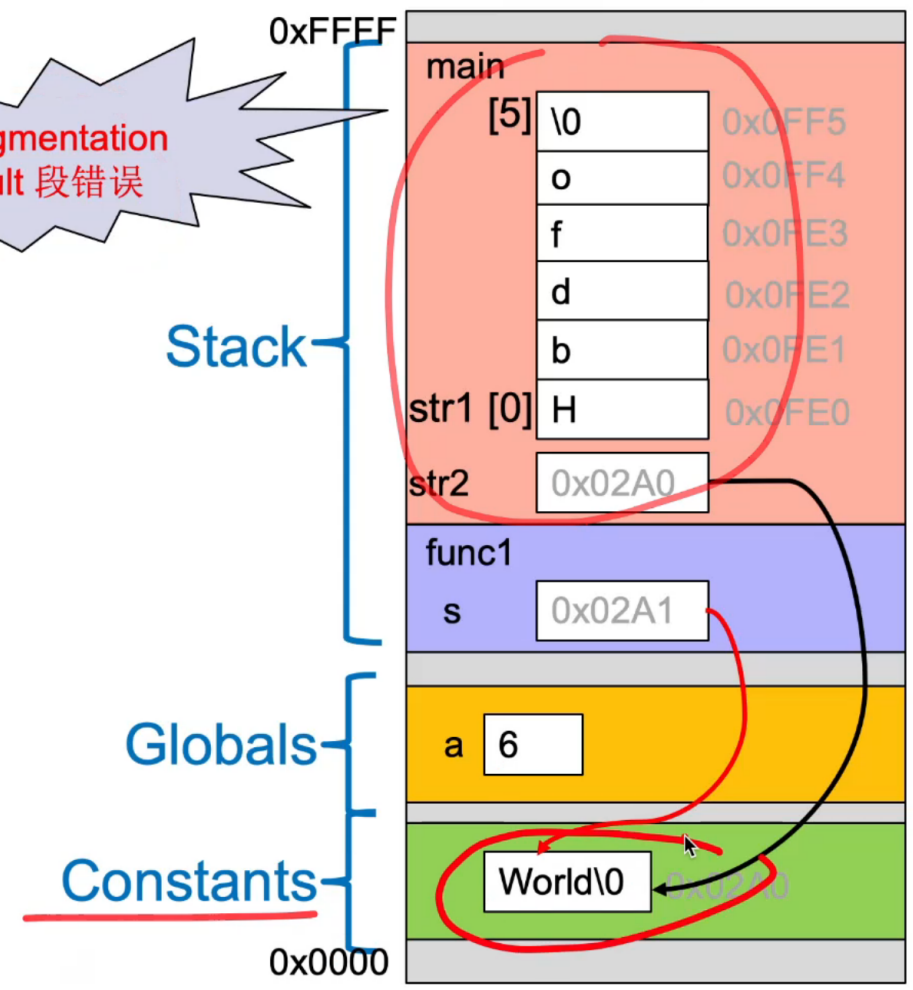
字符串

```
int a = 1;

void func1(char *s)
{
    s[1] = 'a' + a++;
    *(s+2) = 'a' + ++a;
}

int main(void)
{
    char str1[] = "Hello";
    char *str2 = "World";
    func1(str1);
    func1(str1+1);
    func1(str2+1);
    return 0;
}
```

Segmentation Fault 段错误



文件

假设文件能正确打开和关闭，变量都已正确定义，下面程序段输出 6.100#2.000#

```
fp = fopen("data", "w");
fprintf(fp, "%.3f", 3.141596);
rewind(fp);
fputc('6', fp);
fseek(fp, -2L, SEEK_END);
fputs(" ", fp);
fclose(fp);

fp = fopen("data", "r");
while (fscanf(fp, "%lf", &value) != EOF)
  printf("%.3f#", value);
fclose(fp);
```

注意四舍五入

← 会直接在那个位置替换

利用函数fseek可实现的操作是（）

- A. 改变文件指针fp的值
- B. 文件的顺序读写
- C. 文件的随机读写
- ✓ D. 以上答案均正确

假设文件能正确打开和关闭，变量都已正确定义，运行以下文件读写代码，第一个printf输出 Ynwi

空格

```
fp = fopen("data.txt", "w");
fputs("You know - one loves the sunset, when one is so sad", fp);
fclose(fp);

fp = fopen("data.txt", "r");
ch1 = fgetc(fp);
fseek(fp, 12L, SEEK_SET);
ch2 = fgetc(fp);
fseek(fp, -6L, SEEK_CUR);
ch3 = fgetc(fp);
fseek(fp, -12L, SEEK_END);
ch4 = fgetc(fp);
printf("%c%c%c%c", ch1, ch2, ch3, ch4);
fgets(str, 128, fp);
printf("%s", str);
fclose(fp);
```

全角字符：2L

第三次小测

① 当数组名作为实参时，如int fun(int a[]); int b[10]; fun(b);，可以使用sizeof(a)/sizeof(a[0])计算数组元素个数。 ✗

- □ 数组名作为实参时，退化为指针，传递地址。
- □ sizeof(b)是数组大小，sizeof(a)是指针大小

只有当数组为退化成指针后其 sizeof 为指针大小

② 假设scanf语句执行时输入ABCDE<回车>，能使puts(s)语句正确输出ABCDE字符串的程序段是__。

- A. char s[5]={"ABCDE"}; puts(s);

③ 下列程序段执行后输出的结果是

```
int a[] = { 1, 2, 1 };
```

b[2][2]
b[2][1]
b[2][0]

- 字符数组长度 = 字符串有效长度 + 1

B. char s[5]={'A', 'B', 'C', 'D', 'E'}; puts(s);
- 字符数组，缺少'\0'，不是字符串

C. char *s; scanf("%s", s); puts(s);
- 指针变量s未指向有效内存单元，scanf写入段错误

D. char *s; s="ABCDE"; puts(s);

```
int b[][3] = { 1, 2, 3, 10, 20, 30, 50, 60, 70 };
printf("%d", b[a[1]][-a[2]]++);
```

b[1][2]
b[1][1]
b[1][0]
b[0][2]
b[0][1]
b[0][0]

- b[][3] → b[3][3]
- a[1] = 2, a[2] = 1
- b[2][-1]++ → b[2][-1]
  - b + 2 * sizeof(int[3]) + (-1) * sizeof(int)
  - b[2][0] = 50, b[2][1] = 60, b[2][-1] = b[1][2] = 30

①

```
void strdel(char *s, int i, int n) {  /* 删除s[i]到s[i+n]之间的字符 */
    int t = 0;
    char *p = s;
    while (*s++)  t++;      /* t是字符串s的长度 */
    if (i > t)              /* 删除的起始位置大于字符串长度，"0", 2 → "0" */
        return;
    s = p;                  /* s已指向\0，重新指向首字符 */
    if (s + i + n > s + t)  /* 删除所有尾部字符， "01234", 2, 4 → "01\034" */
        s[i] = '\0';
    else {
        s = s + i;          /* 删除中间内容, "01234567", 2, 4 → "0167\0567" */
        while (*s = *(s + n))
            s++;
    }
```

②

定义变量 `char *p = "array";` 则 `p` 等于（）。

- A. 'a'的地址
- B. 'a'的ASCII码
- C. 数组"array"整体的地址
- D. 数组"array"的全体内容

宏定义

```
#define SUM(a,b) printf(#a " + " #b " = %d\n", ((a)+(b)))
SUM(1 + 2, 3 + 4);  // #a → "a" 字符串化，字符数组
等价：printf("1 + 2 + 3 + 4 = %d\n", ((1 + 2) + (3 + 4)));

#define NAME(n) num ## n
int num3 = 5;
printf("%d", NAME(3));  // ## 记号粘贴操作符
等价： printf("%d", num3);
```

指针作为函数的返回值

①

```
#include <stdio.h>
#define T(c) (c == c == c)

double f1()
{
    int x = 1;
    return x;
}
int main(void)
{
    printf("%.1f\n", f1());
    int x = 1;
    printf("%.1f",x);
    return 0;
}
```

```
C:\Users\吴语若\Desktop\C程序\test\项目2.exe
1.0
0.0
--------------------------------
Process exited after 0.02441 second
请按任意键继续. . .
```

```
#include <stdio.h>
double f2()
{
    return (double)5/2;
}
double f3(int n)
{
    if(n == 1) return 1.0;
    else return 1.0 + 1.0/f3(n-1);
}
int main(void)
{
    printf("%.1f\n", f2());
    printf("%.3f\n", f3(4));
    double x = (double)5/2;
    printf("%lf",x);
    return 0;
}
```

```
C:\Users\吴语若\Desktop\C程序\test\项目2.exe
2.5
1.667
2.500000
--------------------------------
Process exited after 0.02533 seconds with return value 0
请按任意键继续. . .
```

关于return：会向高层级数据类型转换

但高→低不行

②

如果有函数 `char *func(char *p, char ch)`，则下面说法错误的是（）。

- A. 函数返回一个字符指针
- B. 可以通过语句 `return NULL;` 返回函数结果
- C. 可以通过语句 `return -1;` 返回函数结果
- D. 可以通过语句 `return p;` 返回函数结果

④

```
char *match( char *s, char ch1, char ch2 )
{
    char *p1 = "";
    char *p2 = "";
```

第四次小测：

① 若执行fopen函数时发生错误，则函数的返回值是0。

- if ((fp = fopen(…)) != NULL) {
- 返回NULL，NULL就是0

② 对于以下结构和变量声明，赋值表达式不正确的是（）。

```
struct Student {
    long num;
    char name[20];
} st1, st2 = {101, "Tom"}, *p = &st1;
```

- A. st1 = st2　　结构赋值
- B. p->name = st2.name　　数组赋值(数值名常量)
- C. p->num = st2.num　　整型赋值
- D. *p = st2　　结构赋值

数组整体不能直接赋值

- 不正确的赋值或赋初值的方式是（）。
  - A.char str[] = "happy new year";
    - 字符串常量初始化字符数组，字符数组长度 = ？
  - B.char str[64]; str = "happy new year";
    - str是数组名，指针常量，不能赋值
  - C.char *p = "happy new year";
    - 字符串常量的首字符地址 初始化 指针变量p
  - D.char *p = NULL; p = "happy new year";
    - 字符串常量的首字符地址 赋值 指针变量p

③ R4-11　分数 3

下面程序的输出为 ~~45675678~~ 15678678　3 分 。

```
#include <stdio.h>
void change(int *p)
{
    *p = *(p + 3);
}
int main()
{
    int x[8] = { 1, 2, 3, 4, 5, 6, 7, 8 }, n = 0;
    while (n++ < 4)
        change(&x[n]);
    for (n = 0; n < 8; n++)
        printf("%d", x[n]);
    return 0;
}
```

判断时n++　操作时i++

④

- sizeof

```
typedef struct my_struct {
    int x;
    char c;
} my_struct;
sizeof(my_struct) = ? 8
```

取最长数据类型 * 个数

⑤ □细节3：int a[10], *p = a; int fun(int b[10]); 退化成指针

区分内存空间分配: sizeof(a), sizeof(p), sizeof(b)

40　4　4

⑥

```
int n = sizeof(++k);
```

sizeof内不执行

⑦

```
unsigned char x = 255, y = '\1';
switch(!!x) {
    case 0: printf("*0*#"); break;
    case 1:
        switch(y) {
            case 0: printf("*1*#"); break;
            case 1: printf("*2*#"); break;
        }
    default: printf("*3*#");
}
```

外部无break

⑧ (3) 课后作业8：设字符型变量x的值为064，则表达式~x ^ x << 2 & x的值？

A. 0333　B. 333　C. 0x333　D. 020

x = 064 = 00 110 100

(~x) ^ ((x << 2) & x)

x << 2 & x: 11 010 000 & 00 110 100

(~x) ^ (…): 11 001 011 ^ 00 010 000

= 11 011 011 = 0333

⑨ □参数计算顺序为从右往左 (编译器相关)

- int i = 5; fun(++i,++i); // Multiple unsequenced modifications to 'i'

⑩

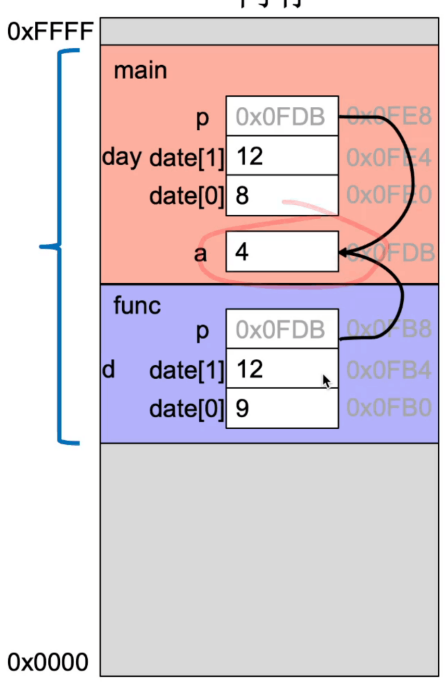
(n>0) ?2.9:1

当n=-1时

会类型转换表达式值为1.0

数组+指针+结构

```
typedef struct date {
    int date[2];
    int *p;
} date;
void func(date d) {
    d.date[0]++;
    (*d.p)++;
}
int main(void) {
    int a = 3;
    date day;
    day.date[0] = 8;
    day.date[1] = 12;
    day.p = &a;
    func(day);
    return 0;
}
```



函数	返回值
fopen	成功：文件指针，失败：NULL
fclose	成功：0，失败：非0
fgetc	成功：读入字符，失败：EOF
fgetc	成功：输出字符，失败：EOF
fputs	成功：最后一个字符，失败：EOF
fgets	成功：字符串，失败：NULL
fscanf	成功：读入参数个数，失败：EOF
fprintf	成功：输出的字符数，失败：EOF
fread	读入元素个数
fwrite	写入元素个数
feof	文件未结束：0，文件结束：1

指针数组 vs. 二维数组

- 多个字符串处理 - 运行时输入

□ 指针数组

```
char *pcolor[10];
for (i = 0; i < 10; i++)
    scanf("%s", pcolor[i]);
```

段错误

□ 二维字符数组

```
char ccolor[10][81];
for (i = 0; i < 10; i++)
    scanf("%s", ccolor[i]);
```

常见错误：pcolor[i]是指针，但未指向有效内存地址 (NULL不是有效内存地址)

具体算法

- 字符大小写转换、数字与数字字符转换、闰年判断 (字符处理)
- 基于数学公式计算π、整数位数、进制转换、素数判断、求最大值、计算Fibnonacci、求水仙数、最大公约数 (循环应用)
  - 最小公倍数：a * b / gcd(a, b)
  - 三个数a, b, c的最大公约数：gcd(a, gcd(b, c))
- 求最大值及下标、删除某个元素 (数组+循环)
- 排序算法：选择法算法、冒泡算法 (数组+循环)
  - 整数/浮点数/字符数组排序
  - 字符串(二维字符数组)排序、结构数组排序
- 查找算法：顺序查找、二分查找 (数组+循环)
- 回文判断、进制转换、字符串压缩、子串删除 (字符串处理)

期末复习

①

假设 a=2,b='2',s="2",下列逻辑表达式中值为1（真）的是 A

A.(s[1]>'a')|| ~(a+b)

B.(b>a) && (s[2]='\0')

C.!(a+1 && b-a)

D.!a>b && ~(a>b)

地址值非对应内容

②

- switch
  - switch 后面的表达式可以是数值型也可以是字符型表达式
  - 若swtich中表达式的值**不是整数则自动取整**

③

已知职工记录描述如下，下列正确赋值方式是（B）

```
struct worker
{
  char name[20];
  char sex;
  struct birth
  {
    int day;
    int month;
    int year;
  }a;
}w,*p=&w;
A.p->name="Li"
B.p->a.year=2000
C.w.day=25
D.w.birth.month=10;
```

数组不能整体三直接赋值

④ **补充：辗转相除法（p106）**

求最大公约数

```
int gcd(int m,int n)
{
  int r,temp;

  if (m<n)
  {
    temp=m;
    m=n;
    n=temp;
  }
  r=m%n;
  while (r!=0)
  {
    m=n;
    n=r;
    r=m%n;
  }
  return n;
}
```

⑤

1. 设有如下定义

```
char a[]={"Thankyou"};
char b[]={'T','h','a','n','k','y','o','u'};
```

用strlen函数求得字符串的长度

则正确的叙述为

A. 数组a和数组b等价

B.数组a和数组b的长度相等

C.数组a的长度大于数组b的长度

D.数组a的长度小于数组b的长度

⑥

访问数组元素的三种方法对比（三种方法都是等价的）

下标法	数组名	指针变量
a[0]/p[0]	*a	*p
a[1]/p[1]	*(a+1)	*(p+1)
……	……	……
a[i]/p[i]	*(a+i)	*(p+i)

题：`int arr[10],*p=arr;` 以下能表示 arr[3]的是(多选) CE

A. (*p)[3] B.*p[3] C.*(arr+3) D.p+3 E.p[3]

⑦ 注意变量初值；scanf读入时不定%d.

可能需要%f.

C大理论复习

1. 函数返回指针：①返回值可以为指针.

②返回值可以为0（NULL）

但不可以为十等非零数字.

2. 在C语言中，读写操作时会进行自动转换的文件是：文本文件（二进制不行）

3. 二进制文件名不能用.txt作为扩展名，否则二进制文件读写函数fread和fwrite将出错. X