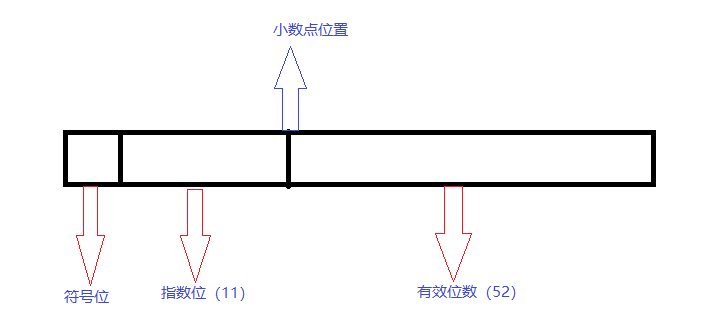
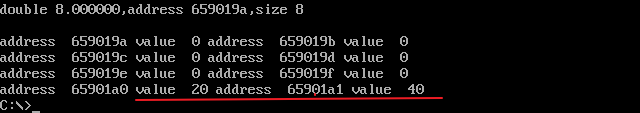
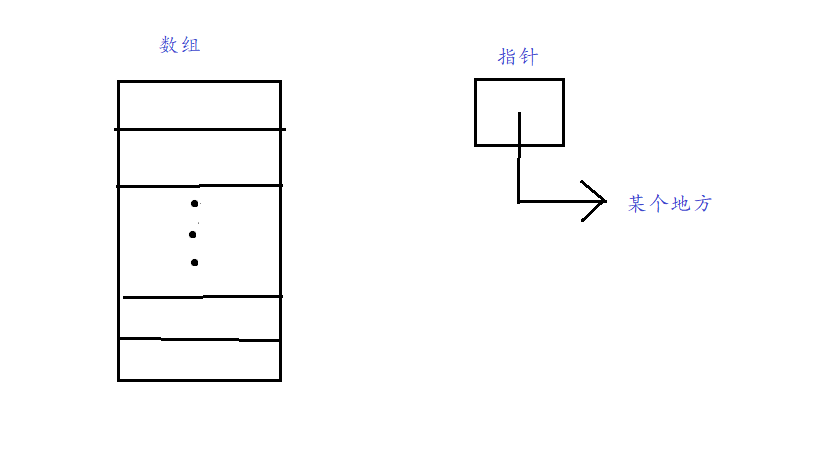
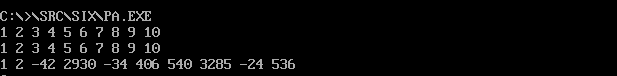
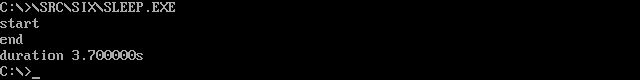
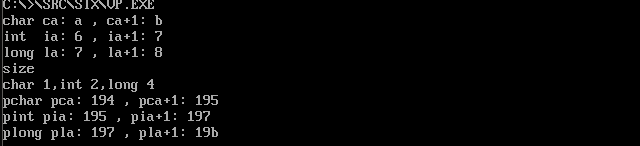
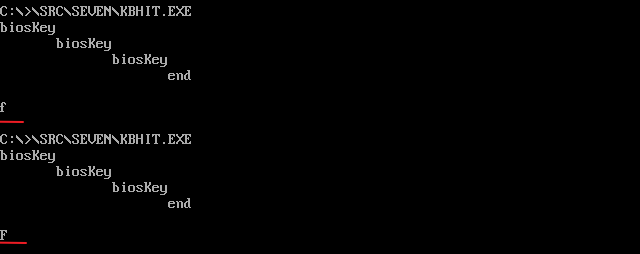
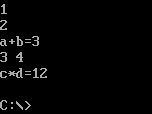
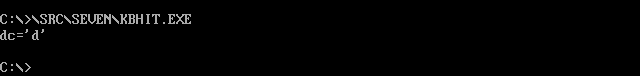
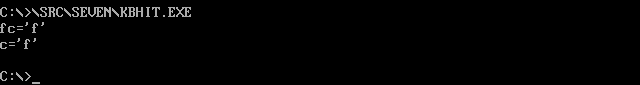
# 综合研究六

* 了解浮点型数据在内存中是怎么存储的。
  + 把双精度的十进制转换为二进制科学计数法然后对应下图存储到内存中
  + 
  + 假设 a=8则其二进制科学计数法表示为 ,双精度浮点数的存储格式为,符号位,指数位,有效位,故对应的二进制表示为0100 0000 0010 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000十六进制为 40 20 00 00 00 00 00 00 00
  + 编写程序验证
    - 程序
    - double da = 8;  
      main() {  
       int i;  
       printf("double %lf,address %lx,size %d\n", da, (long)&da, sizeof(double));  
       for (i = 0; i < sizeof(double); ++i) {  
       if (i % 2 == 0)  
       printf("\n");  
       printf("address %lx ", (long)&da + i);  
       printf("value %x ", \*(char \*)((int)(&da) + i));  
       }  
      }
    - 可以看到下图结果和预测的一致
    - 
* 一维数组和一级指针区别
* 
  + 代码
  + int \*pi;  
    int i;  
    void f();  
    main() {  
     f();  
     for (i = 0; i < 10; i++) {  
     printf("%d ", pi[i]);  
     }  
    }  
    void f() {  
     int ia[10];  
     for (i = 0; i < 10; i++) {  
     ia[i] = i + 1;  
     printf("%d ", ia[i]);  
     }  
     printf("\n");  
     for (i = 0; i < 10; i++) {  
     printf("%d ", \*(ia + i));  
     }  
     printf("\n");  
      
     pi = ia;  
    }
  + 结果
  + 
  + 通过上图可以看到在函数返回后再次打印结果只保存前两个数据
* C中的延时函数
  + sleep（）
  + 测试
  + double start, stop, durationTime;  
    main() {  
     printf("start\n");  
     start = clock();  
     sleep(1 \* 2);  
     stop = clock();  
     printf("end\n");  
     durationTime = ((double)(stop - start)) / 10;  
     printf("duration %lfs", durationTime);  
    }
  + 结果
  + 
* int型变量（存储一般类型（char、int、long等）变量的地址）和指针变量加一操作的区别。
  + 变量加一为值加一，而指针变量加一为指针变量增加变量长度。
  + code
  + char ca = 'a';  
    int ia = 6;  
    long la = 7;  
    char \*pca = &ca;  
    int \*pia = &ia;  
    long \*pla = &la;  
    main() {  
     printf("char ca: %c , ca+1: %c\n", ca, ca + 1);  
     printf("int ia: %d , ia+1: %d\n", ia, ia + 1);  
     printf("long la: %ld , la+1: %ld\n", la, la + 1);  
     printf("size\n");  
     printf("char %d,int %d,long %d\n", sizeof(char), sizeof(int), sizeof(long));  
     printf("pchar pca: %x , pca+1: %x\n", pca, pca + 1);  
     printf("pint pia: %x , pia+1: %x\n", pia, pia + 1);  
     printf("plong pla: %x , pla+1: %x\n", pla, pla + 1);  
    }
  + 结果
  + 

# 综合研究7

* 了解C语言中的输入函数（写程序验证不同输入函数功能上的区别）。
* kbhit()
* main() {  
   char ch;  
   //检查当前是否有键盘输入，若有则返回一个非0值，否则返回0  
   while (!kbhit()) {  
   cprintf("hello world\n");  
   sleep(1);  
   if (kbhit()) {  
   ch = getch();  
   if (ch == 27)//ESE  
   break;  
   }  
   }  
   cprintf("end\n");  
  }
  + 
* bioskey
  + code
  + void f2() {  
     int a;  
     /\* 当用户再循环时不按下任何键时，bioskey(1) 一直为0，则 !bioskey(1)  
     \* 为1，条件成立，继续循环，printf（），直到，用户任意按下一个键时，bioskey(1)不为0，成功退出。  
     \*/  
     a = 1;  
     while (!bioskey(a)) {  
     cprintf("biosKey\n");  
     sleep(1);  
     }  
     cprintf("end\n");  
      
     /\* 当cmd是0，bioskey()返回下一个在键盘键入的值（它将等待到按下一个键）。  
     \* 它返回一个16位的二进制数，包括两个不同的值。当按下一个普通键时，  
     \* 它的低8位数存放该字符的ASCII码,高8位存放该键的扫描码；对于特殊键（如方向键、F1～F12等等），  
     \* 低8位为0，高8位字节存放该键的扫描码。  
     \*/  
     a = 0;  
     printf("\n%c\n", bioskey(0));  
    }
  + 运行结果
  + 
* scanf
  + code
  + void f3() {  
     int a, b, c, d;  
     scanf("%d", &a); /\*输入整数并赋值给变量a \*/  
     scanf("%d", &b); /\* 输入整数并赋值给变量b \*/  
     printf("a+b=%d\n", a + b); /\* 计算a+b的值 \*/  
     scanf("%d %d", &c, &d); /\* 输入两个整数并分别赋值给c、d \*/  
     printf("c\*d=%d\n", c \* d); /\* 计算c\*d的值 \*/  
    }
  + 运行结果
  + 
* getche()
  + code
  + void f4(){  
     char c;  
     c = getche();  
     printf("c='%c'\n", c);  
    }
  + 运行结果
  + 
* getch
  + code
  + void f5() {  
     char c = getch();  
     printf("c='%c'\n", c);  
    }
  + 运行结果 getche()有回显，getch()无回显
  + 
* gets
  + code
  + void f6() {  
     /\* gets() 会读取用户输入的整行内容，包括空格。而 scanf()  
     \* 遇到空格就结束读取，也就是说，使用 scanf()  
     \* 读取的字符串中永远不会包含空格。 \*/  
     char str1[30], str2[30]; /\* 定义两个字符数组 \*/  
     gets(str1);  
     scanf("%s", str2);  
     puts(str1);  
     puts(str2);  
    }
  + 运行结果
  + 