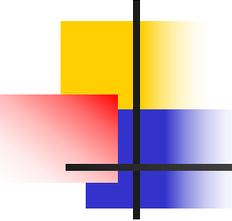


一、选择题

- (1)下列选项中，合法的C语言关键字是：**C**
A) VAR B) integer C) default D) public
- (2)以下选项中不属于C语言的数据类型是：**D**
A)整型 B)实型 C)双精度型 D)复数型
- (3)在C语言中，不正确的short类型（假设存储单元数为2个bytes）的常数是：**B**
A) 123 B) 32768 C) 037 D) 0xAF
- (4)以下选项中合法的实型常数是：**A**
A) .58 B) E-3 C) .E2 D) 1.3E



一、选择题 (cont'd)

(5)下面哪种运算的优先级最高: **A**

- A) ! B) != C) || D) &&

(6) $7/9*9$ 在C语言中的计算结果是: **C**

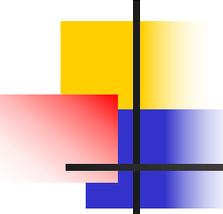
- A) 1 B) 0.08642 C) 0 D) 以上都不是

(7) C语言程序的基本单位是(): **A**

- A) 语句 B) 程序行 C) 函数 D) 字符

(8)以下哪一个是main()的正确声明: **D**

- A) `int main()`
B) `int main(int argc, char *argv[])`
C) A和B都是不正确的
D) A和B都是正确的



一、选择题 (cont'd)

(9) 以下叙述中正确的是：**D**

- A) 可以在一个函数中定义另一个函数
- B) `main()`函数必须放在其它函数之前
- C) 所有被调用的函数一定要在调用之前进行定义
- D) 以上皆不正确

(10) 表达式`!(1&&1||1&&0)`的值是：**C**

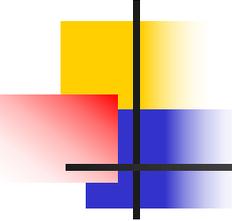
- A) 错误的表达式
- B) `True/1`
- C) `False/0`
- D) 不确定

(11) 有数组定义`int a[2][2]={{1},{2,3}}`;则`a[0][1]`的值为：**A**

- A) 0
- B) 1
- C) 2
- D) 3

(12) 下列描述中不正确的是：**C**

- A) 字符型数组中可以存放字符串
- B) 可以对字符型数组进行整体输入、输出
- C) 可以对整型数组进行整体输入、输出
- D) 不能在赋值语句中通过赋值运算符`=`对字符型数组进行整体赋值



一、选择题 (cont'd)

(13) 这个循环语句会执行多少次: `for (int x=0; x=3; x++)`: **D**

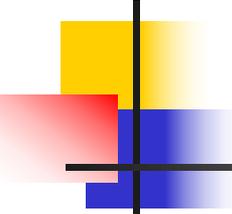
A) 1次都不 B) 3次 C) 4次 D) 死循环

(14) 设有如下定义: `int x=1, y=-1`, 则语句: `printf("%d\n", (x--&++y))`;
的输出结果是: **B**

A) 1 B) 0 C) -1 D) 2

(15) 十进制数1384转换成十六进制数为: **B**

A) 567 B) 568 C) D84 D) D54



一、选择题 (cont'd)

(16)以下程序的输出结果是: **A**

A)##*##*# B)##### C)***** D)*##*##*

```
#include<stdio.h>
int main( )
{ int i;
  for(i=1;i<6;i++) {
    if(i%2){
      printf("#");
      continue;
    }
    printf("*");
  }
  printf("\n");
}
```

一、选择题 (cont'd)

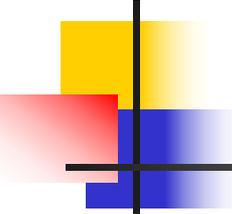
(17) 以下程序的输出结果是: **A**

A) 编译不通过, 无输出 B) aceg C) acegi D) abcdefghi

```
int main( )
{ int i;
  for(i='A';i<'I';i++, i++)
    printf("%c", i+32);
  printf(" \n");
}
```

(18) 以下程序的运行结果是: **B** A) 6 B) 6789 C) '6' D) 789

```
#include<stdio.h>
int main( )
{ char a[10]={'1', '2', '3', '4', '5', '6', '7', '8', '9', 0}, *p;
  int i;
  i=8;
  p=a+i;
  printf("%s\n", p-3);}
```

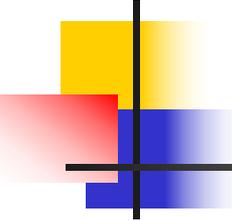


一、选择题 (cont'd)

(19) 以下程序的输出结果是：**B**

A) 4 2 1 1 B) 0 0 0 8 C) 4 6 7 8 D) 8 8 8 8

```
main( ) {
    char s[]="12134211";
    int v[4]={0,0,0,0},k,i;
    for(k=0;s[k];k++) {
        switch(s[k]) {
            case '1':i=0;
            case '2':i=1;
            case '3':i=2;
            case '4':i=3;
        }
        v[i]++;
    }
    for(k=0;k<4;k++)
        printf("%d ",v[k]);
}
```



一、选择题 (cont'd)

(20)运行以下程序将会出现: **D**

- A) 输出0123..99 B) 输出0123...100 C) 输出0123...101 D) 不确定

```
main( )  
{  
    int x;  
    while(x<100) {  
        printf("%d",x);  
        x++;  
    }  
}
```

二、填空题

- (1) `#include <filename.h>` 和 `#include "filename.h"` 有什么区别？：
前者编译器只会到系统指定目录下搜索头文件；后者编译器会先在工作目录下查找头文件，如果找不到，则到系统指定目录下搜索。

- (2) 下列程序的输出结果是：_____
- ```
#include <stdio.h>
main()
{ char ch[]="abc",x[3][4];int i;
 for(i=0;i<3;i++)strcpy(x[i],ch);
 for(i=0;i<3;i++)printf("%s",&x[i][i]);
 printf("\n");
}
```
- abcbcc

## 二、填空题 (cont'd)

(3) 下列程序的输出结果是: \_\_\_\_\_

```
#include<stdio.h>
```

```
int main()
```

```
{ int a=1,b;
```

```
 for(b=1;b<=10;b++) { /* b=1, a=6; b=2, a=3; b=3, a=8; b=4 */
```

```
 if(a>=8)break;
```

```
 if(a%2==1) {
```

```
 a+=5;continue;
```

```
 }
```

```
 a-=3;
```

```
 }
```

```
 printf("%d\n",b);
```

```
}
```

4

# 三、程序改错题

```
int binsearch(int x, int v[], int n){
 int low, high, mid;
 low=1;
 high=n;
 mid=(high-low)/2;
 while(low<high && x!=v[mid]) {
 if (x<v[mid])
 high=mid-1;
 else
 low=mid+1;}
 if (x==v[mid])
 return mid; /* found match */
 else
 return -1; /* no match */
}
```

```
int binsearch(int x, int v[], int n) {
 int low, high, mid;
 low=0;
 high=n-1;
 mid=(low+high)/2;
 while(low<=high && x!=v[mid]) {
 if(x<v[mid])
 high=mid-1;
 else
 low=mid+1;
 mid=(low+high)/2;
 }
 if(x==v[mid])
 return mid;
 else
 return -1;
}
```

## 三、综合题(略)

- 输入一个自然数 $N$  ( $N > 1$ )，输出它的质因子分解式。如 $n = 8$ ，程序输出 $8 = 2 * 2 * 2$
- 如果有 $n$ 个棋子，甲、乙两方轮流取棋子，每方至少取一个棋子，最多取 $m$ 个棋子。谁最后取剩下的一个棋子，谁就输，如果甲方先取，请给出甲方必赢的算法。(即：无论乙怎么下，甲都能赢)
- 设有 $N$ 个白石子和 $N$ 个黑石子，开始时排成一行，中间有一个空位置。如 $N = 3$ ，三个白石子和三个黑石子排成如下：**WWW**  
**BBB**。要求按以下的规则将他们移动，变成 $N$ 个黑石子和 $N$ 个白石子排成一行，中间有一个空位置：**BBB WWW**。规则限定每次只能移动一个石子，并要求向着他的目标方向移动，或者将一个石子移动到相邻的空位置，或者跳过一个不同颜色的石子进入空位置。设计移动的策略，并编写程序，输出每次移动后的状态。