1 C 语言函数与存储部分练习参考题

1.1 填空题

1. 在程序的整个运行期间都有效的变量包括（*static* 修饰的变量）以及（全局变量）。

2. 在 C 语言中，如果要进行动态存储管理（亦即管理堆空间），通常使用 `malloc()` 和 `free()` 两个函数。它们在头文件（*stdlib.h*）中定义。

3. 代码块 `int *p = malloc(sizeof(int));`
`p = malloc(sizeof(int) * 2);` 所犯的错误为（内存泄漏），也可以写做（前一个语句分配的空间，因为后一个语句的执行而丢失）。

1.2 选择题（每题恰好有一个选项符合要求）

1. 以下关于 heap 空间和 stack 空间说法，不恰当的是（*D*）。

   A 每个程序有自己的 stack 空间

   B 所有程序共享 heap 空间

   C stack 空间比 heap 空间小得多

   D stack 空间的分配和回收自动进行

2. 以下关于 heap 空间，stack 空间和 static 空间的说法，不恰当的是（*D*）。

   A heap 空间的分配和回收比 stack 空间慢

   B static 空间中的变量在程序的整个运行期间有效

   C 函数调用开始，相应的 stack 空间被分配

   D 函数调用结束，相应的 heap 空间被自动回收

3. 以下关于 heap 空间，stack 空间和 static 空间的说法，不恰当的是（*B*）。

   A 全局变量存放在 static 空间

B static 修饰的变量所占的空间在定义它的函数运行结束之后被回收

C 函数被调用时分配的 heap 空间，可以在该函数执行完毕前被回收

D 一个函数分配的 heap 空间，可以被另外一个或多个函数回收

4. 考虑以下代码段

```c
#include <stdio.h>
#include <stdlib.h>
int n;
int main(){
   double x;
   double *p = malloc(3 * sizeof(double));
}
```

则下列说法不恰当的是 (*D*) 。

A `n` 存放在 static 空间

B `x` 存放在 stack 空间中

C `p` 存放在 stack 空间中

D heap 空间分配给程序的内存块可以是不连续的

5. 假设`sizeof(int)`的值为 4，`sizeof(float)`的值为 8，每个指针变量所占的空间大小为 8 个字节。考虑程序段，

```c
#include <stdlib.h>
int main(){
   int a; float *p = malloc(2*sizeof(float));
}
```

则以下说法不恰当的是 (*B*) 。

A stack 空间存放程序中的 2 个变量

B stack 空间存放程序中的 1 个变量，static 空间中存放程序中的 1 个变量

C 上述代码中的变量在 stack 空间中占用 12 个字节

D 上述代码在 heap 空间中最多时占用 16 个字节

6. 考虑以下代码段

I. `malloc(sizeof(int));` // 假设`sizeof(int)` 的值为 4

II. `malloc(2 * sizeof(short));` // 假设`sizeof(short)` 的值为 2

III. `malloc(4 * sizeof(char));` // 假设`sizeof(char)` 的值为 1

IV. `malloc(4);` //

则以下说法恰当的是 (*A*) 。

A 四个代码段完全等价

B 代码段（I）和（II）不等价

C 代码段（I）和（III）不等价

D 代码段（III）和（IV）不等价

7. 以下不可以作为假的是 (*C*) 。

A `malloc()` 函数返回的 `NULL`

B `0`

C `'0'`

D `'\0'`

8. 假设`sizeof(long)` 的值为 8，并且每个指针占用 8 个字节。考虑以下代码段

```
#include <stdio.h>
#include <stdlib.h>
long n;
int main(){
   long *a[4];
   for(long i = 0; i < 3; i++)
      a[i] = malloc(sizeof(long) * 4);
}
```

则下列说法不恰当的是 (*B*)

A heap 空间前后一共被分配 32 个字节给程序

B heap 空间先后分配给程序的内存块连接在一起

C stack 空间最多的时候分配 40 个字节给程序中的变量

D static 空间分配给程序 8 个字节

9. 假设已经引入头文件 stdlib.h，考虑以下代码段

I. `int n; int *p = (int*)malloc(sizeof(int)); p = &n;`

II. `int* f(){return (int*)malloc(sizeof(int));}...int n; int *p = f(); p = &n;`

III. `int n; int *p = (int*)malloc(sizeof(int)); int *q = p; p = &n;`

下来说法恰当的是 (*D*) 。

A 只有代码段（I）发生内存泄漏

B 只有代码段（II）和（III）发生内存泄漏

C 三段代码都发生内存泄漏

D 只有代码段（I）和（II）发生内存泄漏

10. 假设已经引入头文件 stdlib.h，考虑以下代码段

I. `int *ptr = (int *)malloc(sizeof(int)); int *qtr = ptr; free(ptr); (*qtr) = 5;`

II. `char *p; {char ch = 'A'; p = &ch;} (*p) = 'a';`

III. `int *fun(){static int x = 5; return &x;}...int *p = fun(); (*p) = 5;`

IV. `int *fun(){int x = 5; return &x;}...int *p = fun(); (*p) = 5;`

则以下说不恰当的是 (*B*) 。

A 代码段（II）发生产生悬浮指针

B 代码段（I）和（III）产生悬浮指针

C 代码段（IV）产生悬浮指针

D 代码段（I）和（II）产生悬浮指针

11. 以下不能被传入函数的是 (*A*) 。

A 数组

B 指针

C 整型

D 浮点型

12. 考虑以下代码段

```
1  #include <stdio.h>
2  int sqr(int x)
3  {
4     x = x * x;
5     printf("x val: %d, addr of x: %p", x, &x);
6     return x;
7  }
8  int main()
9  {
10    int t = 10;
11    printf("%d", sqr(t));
12    printf("t val: %d, addr of t: %p", t, &t);
13    return 0;
14 }
```

关于第 5 行和第 12 行，以下说法恰当的是 (*C*) 。

A 两次输出的整型值相同，地址值不同

B 两次输出的整型值不同，地址值相同

C 两次输出的整型值和地址值都不同

D 两次输出的整型值和地址值都相同

13. 考虑以下代码

```
1  #include <stdio.h>
2  int square(int n)
3  {
4    n *= n;
5    printf("n val: %d, addr of n: %p\n", n, &n);
6    return n;
7  }
8  int main()
9  {
10   int x = 8;
11   printf("%d\n", square(x));
12   printf("x val: %d, addr of x: %p", x, &x);
13   return 0;
14 }
```

关于第 5 行和第 12 行，以下说法恰当的是 (*B*) 。

A 两次输出的整型值相同，地址值不同

B 两次输出的整型值不同，地址值相同

C 两次输出的整型值和地址值都不同

D 两次输出的整型值和地址值都相同

14. 下列程序输出的结果是 (*A*) 。

```
1  void operate(int *p, int* q)
2  {
3     int temp = *p; *p = *q; *q = temp;
4  }
5  int main()
6  {
7     int x = 3, y = 5;
8     operate(&x, &y);
9     print("%d,%d,", x, y);
10    operate(&y, &x);
11    print("%d,%d", x, y);
12    return 0;
13 }
```

A 5，3，3，5

B 3，5，5，3

C 3，5，3，5

D 5，3，5，3

15. 下列程序输出的结果是 (*C*) 。

```
1  void operate(int p, int q)
2  {
3     int temp = p; p = q; q = temp;
4  }
5  int main()
6  {
7     int x = 3, y = 5;
8     operate(x, y);
9     print("%d,%d,", x, y);
10    operate(y, x);
11    print("%d,%d", x, y);
12    return 0;
13 }
```

A 5，3，3，5

B 3，5，5，3

C 3，5，3，5

D 5，3，5，3

16. 考虑以下代码，假设用户输入“ABC”。

```
#include <stdio.h>
#include <ctype.h>
void print_upper1(char *string){
   for(int t = 0; string[t]; ++t){putchar(string[t] + 32);}
}
void print_upper2(char *string){
   for(int t = 0; string[t]; ++t){string[t] = string[t] + 32;
        putchar(string[t]);}
}
int main(){
   char s[80];
   gets(s);
   print_upper1(s);
   puts(s);
   print_upper2(s);
   puts(s);
   return 0;
}
```

上述程序输出（*A*）。

A `abcABCabcabc`

B `abcabcabcabc`

C `abcABCABCabc`

D `ABCABCabcabc`

17. 考虑以下代码，假设生成的可执行文件的名字为 `run`，并且用户在 Windows 操作系统下运行程序时输入`.\run this file now`。

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
   printf("%d%s%c%c", argc, argv[1], *argv[2], argv[3][2]);
   return 0;
}
```

则以上代码输出 (*B*) 。

A `3thisfw`

B `4thisfw`

C `3thisfilenow`

D `4thisfilew`

18. 考虑以下代码段

```
int f1(int a, int b){
   if(a > b) return 1;
   else if(a < b) return 0;
}
int f2(int n){
   for(int i = 2; i < n; i++)
      if(!(n % i)) return i;
}
```

则 (*D*) 。

A 函数 `f1()` 存在返回漏洞，函数 `f2()` 不存在返回漏洞

B 函数 `f1()` 不存在返回漏洞，函数 `f2()` 存在返回漏洞

C 函数 `f1()` 和函数 `f2()` 都不存在返回漏洞

D 函数 `f1()` 和函数 `f2()` 都存在返回漏洞

19. 考虑以下代码，则以下说法恰当的是 (*B*) 。

```c
#include <stdio.h>
int mul(int a, int b) {return a * b;}
int main(){
   int x, y, z;
   x = 10; y = 20;
   z = mul(x, y); /* I */
   printf("%d,%d", z, mul(x,y)); /* II */
   mul(x, y); /* III */
   printf("%d,%d", z, mul(x,y));
   return 0;
}
```

A 删除（I）对程序的输出结果不影响，删除（III）对程序的输出结果有影响

B 删除（I）对程序的输出结果有影响，删除（III）对程序的输出结果不影响

C 删除（I）和（III）对程序的输出结果都不影响

D 删除（I）和（III）对程序的输出结果都不影响

20. 考虑以下

```c
#include <stdio.h>
char *f(char c, char *s){while(c != *s && *s) s++; return s;}
int main()
{
   char s[80] = "dcba";
   char *p1, *p2, *p3;
   p1 = f('a', s);
   p2 = f('c', s);
   p3 = f('e', s);
   printf("%p,%p,%p\n", p1, p2, p3);
   return 0;
}
```

则下列有可能分别是三个输出值的后 6 位的是 (*A*) 。

A `61FDA3,61FDA1,61FDA4`

B `61FDA1,61FDA3,61FDA4`

C `61FDA3,61FDA1,61FDA2`

D `61FDA1,61FDA3,61FDA2`

21. 下列程序输出的结果是 (*A*) 。

```
void operate(int n, int *p, int* q)
{
   for(int i = 2; ; i++)
   {
      if(i * i > n) break;
      if(n % i == 0)
      {
         (*p) = i;
         (*q) = n / i;
         return;
      }
   }
   (*p) = 1; (*q) = n;
}
int main()
{
   int x, y;
   operate(15, &x, &y);
   print("%d,%d,", x, y);
   operate(18, &y, &x);
   print("%d,%d", x, y);
   return 0;
}
```

则输出是 (*B*) 。

A 3, 5, 2, 8

B 3, 5, 8, 2

C 5, 3, 2, 8

D 5, 3, 8, 2