1. 在 win32 x86 模式下，int *p; int **pp; double *q; 请说明 p、pp、q 各占几个字节的内存单元。

参考答案

不管是什么类型的指针，也不管是几重指针，它们存放的是一个物理地址。所以，p、pp、q 都是占 4 个字节的内存单元。

2. 常量 1、1.0、“1”的数据类型是什么？

参考答案

1	int
1.0	double
“1”	const char *

3. 语句：short int a[10]; short int *p = a; sizeof(a)等于 sizeof(p)吗？为什么？

参考答案

a 表示包含 10 个 short int 的缓冲区，p 变量存贮的是 a 的地址。

sizeof(a) = 20

sizeof(p) = 4 (win32 x86)

4. 若给出声明：

```
char c, *pc;
const char cc = 'a';
const char *pcc;
char *const cpc = &c;
const char * const cpcc = &cc;
char *const *pcpc;
```

则下面的赋值哪些是合法的？哪些是非法的？为什么？

```
(1) c = cc;              (10) *pc = "ABCD"[2];
(2) cc = c;              (11) cc = 'a';
(3) pcc = &c;            (12) *cpc = *pc;
(4) pcc = &cc;           (13) pc = *pcpc;
(5) pc = &c;             (14) **pcpc = *pc;
(6) pc = &cc;            (15) *pc = **pcpc;
(7) pc = pcc;            (16) *pcc = 'b';
(8) pc = cpcc;           (17) *pcpc = 'c';
```

(9) cpc = pc;　　(18) *cpcc = 'd';

参考答案：

(1) 合法，c不是const类型的变量，可以赋值。

(2) 非法，cc是const类型的变量，不能赋值。

(3) 合法，pcc不是const类型的指针变量，可以指向非const类型的字符变量。

(4) 合法，pcc不是const类型的变量，赋值后指向的cc的类型为const char，同其要求指向的类型一致。

(5) 合法，pc 不是 const 类型的指针变量，赋值后指向的 c 的类型为 char，同其要求指向的类型一致。

(6) 非法，pc要求指向char类型的变量，不能用const char*类型的&c赋值。

(7) 非法，pc要求指向char类型的变量，不能用指向const char*类型的pcc赋值。

(8) 非法，pc要求指向char类型的变量，不能用指向const char*类型的cpcc赋值。

(9) 非法，cpc是const类型的变量，不能赋值。

(10) 合法，pc指向的是非const类型的变量，可以赋值，等价于*pc='C'。

(11) 非法，cc是const类型的变量，不能赋值。

(12) 合法，cpc指向的是非const类型的变量，可以赋值。

(13) 合法，pc是非const类型的指针变量，可以用char *类型的值*pcpc赋值。

(14) 合法，**pcpc代表的是非const类型的字符变量，可以任何字符类型的值赋值。

(15) 合法，*pc代表的是非const类型的字符变量，可以任何字符类型的值赋值。

(16) 非法，*pcc代表的字符是const类型的字符变量，不能赋值。

(17) 非法，*pcpc代表的是const类型的指针变量，不能赋值。

(18) 非法，*cpcc 代表的是 const 类型的只读变量，不能赋值。

5. C按优先级和结合性解释类型，下述声明是什么意思?

(1) typedef void VF_PC_RI(char*, int &);

(2) typedef VF_PC_RI* P_VF_PC_RI;

(3) typedef int &RIFFII(int, int);

(4) extern VF_PC_RI funca;

(5) extern P_VF_PC_RI ptra;

(6) extern void func1 (P_VF_PC_RI *);

(7) extern P_VF_PC_RI func2 (int c);

(8) P_VF_PC_RI func3 (P_VF_PC_RI a);

(9) typedef void (*(**VF_PA_P_PF_V(void))[ ]) (const int);

(10) int *(*p)[4][2];

参考答案：

(1) 定义一个名为VF_PC_RI的类型，该类型定义了一个参数为(char*, int &)没有返回值的函数。

(2) 定义一个名为P_VF_PC_RI的类型，该类型定义了一个指向VF_PC_RI类型的指针。

(3) 定义一个名为RIFFII的类型，该类型定义了一个参数为(int ,int)返回一个引用的函数，该引用引用一个

整型量。

(4) 说明一个类型为 VF_PC_RI 的函数 funca。

(5) 说明一个类型为P_VF_PC_RI的指针变量ptra。

(6) 说明一个没有返回值的函数func1，该函数的参数是一个指向P_VF_PC_RI类型的指针。

(7) 说明一个参数为int类型的函数func2，该函数的返回值是一个P_VF_PC_RI类型的指针。

(8) 说明一个参数为P_VF_PC_RI类型的函数func3，该函数的返回值是一个P_VF_PC_RI类型的指针。

(9) 定义一个名为 VF_PA_P_PF_V 的类型，该类型定义了一个没有参数的函数，该函数返回一个指针，该指针又指向另一个指针，被指向的指针指向一个数组，数组的每个元素存放一个函数指针，该函数指针指向一个参数为 const int 类型没有返回值的函数。

(10) p 是 1 个指针，这个指针指向 4 个元素的数组，每个数组元素又包含 2 元素，这 2 个元素的值是 int *。或者说，p 是 1 个指针，这个指针指向 1 个二维数组（4 行 2 列），每个数组元素是 1 个指向 int 的指针。

6. 下面g()函数的重载声明和定义是否会导致编译错误？

```
float g(int);
int g(int);
int g(int, int y=3);
int g(int, …);
int i = g(8);
```

参考答案：

(1) 不能定义返回类型仅和 float g(int)不同的函数 int g(int) 。

(2) g(8)在调用时出现二义性，无法确定是调用 int g(int, int y=3)还是 int g(int, …)。

7. 定义函数求 n（n>=1）个 double 类型的数的最大值 double max1(int n, …)。注意：如果编程测试，建议使用 vs2019，并且一定要在 x86 模式（不要在 x64）。有兴趣的同学，可以在 vs2019-x64 下测试，分析为什么得不到正确的结果。

参考答案：

```
double max1(int n, …) {
   double *p = (double *)(&n + 1);   //p 指向第 1 个 double
   double v = p[0];
   for(int k = 1; k < n; k++) {
      if( v < p[k] ) v = p[k];
    }
    return v;
}
```

vs2019-x64：这个函数的指针 p 不能正确地指向第 1 个 double，因为 vs2019 会给每个变量分

配至少 8 个字节。对于 max1 的 int n，会给 n 分配 8 个字节，但 n 本身只占 4 字节，所以要使 p 指向第 1 个整数，必须：p = (double *)((char *)&n + 8); 或者 p = (double *)(&n + 2)。同样的道理，对于 x86 模式，一个函数 f(char c, short n, int m)，vs2019 会给 c、n、m 都分配 4 字节（虽然 c 实际只占 1 字节、n 只占 2 字节）。