目录

1 三种基本表达结构

1.1 顺序结构

```
int tmp;
tmp = 5;
printf("tmp = %d\n",tmp);
```

1.2 选择结构

```
int tmp = 10;
if (tmp >= 5)
    printf("tmp >= 5\n");
else if (tmp > 3)
    printf("3 < tmp < 5\n");
else
    printf(" tmp < 3\n");
```

1.2.1 逻辑运算

1. 与运算

&&		
a	b	ans
1	1	1
1	0	0
0	1	0
0	0	0

(a) 总结

- 与 1 保持不变
- 与 0 变零

2. 或运算

竖线		
a	b	ans
1	1	1
1	0	1
0	1	1
0	0	0

(a) 总结

- 或 1 变 1
- 或 0 不变

3. 非运算

!	
a	ans
0	1
1	0

4. 异或运算

^		
a	b	ans
1	1	0
1	0	1
0	1	1
0	0	0

(a) 总结

- 异或 1 取反
- 异或 0 不变

5. 优先级 与运算 > 或运算 > 非运算

1.3 循环结构

1.3.1 for 循环

```
for (初始化;判断条件;每次循环结束后更新){
    执行语句;
}
```

1.3.2 while 循环

```
初始化;
while (判断条件){
    执行语句;
    每次循环结束后更新;
}
```

1.3.3 中断

1. break 语句 break 跳出当前最近的一层循环

```
for (int i = 0 ;i< 5 ;i++) {
    for (int j = 0 ;j< 5 ;j++){
        if (j==3)
            break;
    }
    int j=0;
    while (j<5){
        if (j==3)
            break;
        j++;
    }
    printf("%d\n",j);
}
```

2. continue 语句

   continue 跳到循环开头

```
for (int i = 0 ;i< 5 ;i++) {
    for (int j = 0 ;j< 5 ;j++){
        if (j==3)
            continue;
    }
    int j=0;
    while (j<5){
        if (j==3){
            j++;
            continue;
        }
        j++;
    }
    printf("%d\n",j);
}
```

2 数组

2.1 一维数组

```
int a[5];
//数组输入
for (int i = 0 ;i< 5 ;i++)
    scanf("%d",&a[i]);
//数组输出
for (int i = 0 ;i< 5 ;i++)
```

```
    printf("%d\n",a[i]);
```

2.1.1 注意事项

- 声明数组

```
int a[100];
const int maxn = 100;
int b[maxn];
//错误
int num = 100;
int c[num];
```

- const 关键字加 const 关键字的变量不能够再改变。如果改变编译器会发生报错。

2.2 多维数组

```
const int maxn = 100;
int num2[maxn][maxn];
int num3[maxn][maxn][maxn];
for (int i = 0;i < maxn ;i++)
    for (int j = 0 ;j < maxn ;j++)
        scanf("%d",&num2[i][j]);
for (int i = 0 ;i< maxn ;i++ ){
    for (int j = 0 ;j< maxn; j++)
        printf("%6d ",num2[i][j]);
    printf("\n");
}
```

3 字符串

3.1 输入

3.1.1 scanf(通过空格与换行符分割字符串)

```
const maxn = 100;
char s[maxn];
scanf("%s",s);
```

3.1.2 gets(通过换行符分割字符串)

```
const maxn = 100;
char s[maxn];
gets(s);
```

3.2 输出

```
char s[1100] = "abc efg\nabc efg\n";
printf("%s\n",s);
\\等价于
for (int i = 0 ;i< strlen(s); i++)
    printf("%c",s[i]);
printf("\n");
\\等价于
for (int i = 0 ; s[i] != 0 ; i++)
    printf("%c",s[i]);
printf("\n");
```

3.3 字符串长度

字符串结束标志'\0' ASCII 码为 0

```
#include <cstring>
const maxn = 1100;
char s[maxn];
scanf("%s",s);
int lenS = strlen(s);
\\等价于
for (lenS = 0 ;s[lenS]!=0 ;lenS++);
```

3.4 ASCII 码表

字符以 ASCII 码的形式存放在计算机中, 换句话说就是以数字的方式存放在计算机中。 ASCII 码表链接

- a~z 之间的 ASCII 码是连续的, 并且是递增的
- A~Z 之间的 ASCII 码是连续的, 并且是递增的
- 0~9 之间的 ASCII 码是连续的, 并且是递增的

```
for (int i = 0; i< 26 ;i++)
   printf("%c",'a'+i);
```

3.5 字符串比较与字典序

3.5.1 字符串比较

```
#include <cstring>
#include <algorithm>
const int maxn = 110;
char s1[maxn],s2[maxn];
scanf("%s%s",s1,s2);
```

```
int ans = strcmp(s1,s2);
//等价于
int strcmp_copy(char s1[], char s2[]){
    int lens1 = strlen(s1);
    int lens2 = strlen(s2);
    int lens = min(lens1,lens2);
    int ans = 0;
    for (int i = 0 ;i<= lens && ans == 0 ;i++)
        ans += s1[i] - s2[i];
    return ans;
}
```

3.5.2 strcmp 用法

1. 输入两个字符串

   ```
   strcmp(s1,s2);
   ```

2. 返回值

   - >0 表示 s1 字典序大于 s2
   - =0 表示 s1 字典序等于 s2 (两个字符串完全相同)
   - <0 表示 s1 字典序小于 s2

3.6 举例

3.6.1 删除多余空格

```
void del_spare_space(char s[]){
   int lens = strlen(s);
   for (int i = 0, j = 0 ; i<=lens && j<= lens ;i++ ,j++){
       while (i!=0 && s[i] == ' ' && s[i-1] == ' ')
           i++;
       s[j] = s[i];
   }
}
```

4 函数

4.1 函数的基本结构

```
返回类型 函数名(输入变量1,输入变量2,...){
    return 返回值;
}
```

4.2 输入变量

4.2.1 取址符作用

- 加上取址符，函数的输入变量与真实的输入变量共享同一份地址
- 不加取址符，函数的输入变量为真实的输入变量的拷贝（复制）。

```
void add1(int a, int b,int c){
    c = a + b;
}
void add2(int a, int b, int& c){
    c = a + b;
}
int a = 1, b = 2;
int c1 = 0 ,c2 = 0;
add1(a,b,c1);
printf("c1=%d\n",c1);
add2(a,b,c2);
printf("c2=%d\n",c2);
```

4.2.2 输入数组

函数中输入的数组与真实输入的数组是共享地址的。在函数中修改输入数组，会改变真实输入的数组。

1. 一维数组

```
//输入的数组
const int maxn = 110;
int num[maxn];
//输入一维数组---函数定义方式1
void add(int a[]);
//输入一维数组---函数定义方式2
void add(int a[maxn]);
//输入一维数组---函数定义方式3
void add(int *a);
```

2. 多维数组

```
//输入的数组
const int maxn = 110;
int num[maxn][maxn];
//输入二维数组---函数定义方式1
void add(int a[maxn][]);
//输入二维数组---函数定义方式2
void add(int a[maxn][maxn]);
```

```
//输入二维数组---函数定义方式3
void add(int **a);
```

3. 举例

```
void add(int a[], int& c){
    c = a[0] + a[1];
}
int num[2];
int c;
scanf("%d%d",&num[0],&num[1]);
add(num,c);
printf("%d\n",c);
```

4.3 递归调用

函数自己调用自己。

```
int jc(int n){
    if (n<0) return -1; //-1表示输入有问题
    if (n==1 || n==0) return 1; // 0和1的阶乘为边界情况
    return n*jc(n-1); // n!=n*(n-1)!
}
```

5 结构体

5.1 结构体定义的基本结构

```
const int maxn = 1e4;
struct BigNum{
    int num[maxn];
    int totl;
    void init(){
       totl = 0;
       for (int i = 0 ;i< maxn ;i++)
           num[i]=0;
    }
};
```

5.2 结构体使用

```
//a表示大整数 123456789
BigNum a;
a.init();
```

```
a.totl = 9;
for (int i = 9 ;i>=1 ;i--)
    a.num[9-i] = i;
```

5.3 例子

5.3.1 大整数

```
const int maxn = 1e4;
struct BigNum{
    int num[maxn];
    int totl;
    void init(){
       totl = 0;
       for (int i = 0 ;i< maxn ;i++)
           num[i]=0;
    }
    void int2BigNum(int a){
        totl = 0;
        while (a){
            num[totl++] = a%10;
            a /= 10;
        }
    }
    void print(){
       for (int i=totl-1; i>=0 ;i--)
           printf("%d",num[i]);
       printf("\n");
    }
};
int main(){
    BigNum a;
    a.init();
    int b = 123456789;
    a.int2BigNum(b);
    a.print();
}
```

5.3.2 二维坐标排序

```
#include <cstdio>
#include <algorithm>
using namespace std;
struct point{
```

```
   int x;
   int y;
};
bool cmp(const point &a, const point &b){
    if (a.x<b.x)
       return true;
    if (a.x == b.x && a.y<=b.y)
       return true;
    return false;
}
int main (){
    point node[110];
    int n;
    scanf("%d",&n);
    for (int i = 0 ;i<n; i++)
        scanf("%d%d",&node[i].x,&node[i].y);
    sort(node,node+n,cmp);
    for (int i = 0 ;i<n; i++)
        printf("%d %d\n",node[i].x,node[i].y);
}
```

6 计算机中程序执行的基本流程与原理

7 模块化编程思想