strncpy(a,b,5); a[5]='\0';
char a[10]; memset(a,'#',sizeof(a));a[10]='\0';
#C：
char st[100];
1. 字符串长度
  strlen(st);
2. 字符串比较
  strcmp(st1,st2);
  strncmp(st1,st2,n); 把 st1,st2 的前 n 个进行比较。
3. 附加
  strcat(st1,st2);
  strncat(st1,st2,n); n 表示连接上 st2 的前 n 个给 st1，在最后不要加'\0'。
4. 替换
  strcpy(st1,st2);
  strncpy(st1,st2,n); n 表示复制 st2 的前 n 个给 st1，在最后要加'\0'。
5. 查找
  where = strchr(st,ch) ch 为要找的字符。返回 char 类型的指针
  where = strspn(st1,st2); 查找字符串。从 st1 开头开始与 st2 不匹配的下标，整形。
  where = strstr(st1,st2); 查找子串，若 st2 是 st1 的子串则返回 char 类型的指针，st2 出现的位置，否则返回 NULL。
#C++队列操作 #include<queue>
C++队列 Queue 类成员函数如下：
back()返回最后一个元素
empty()如果队列空则返回真
front()返回第一个元素
pop()删除第一个元素
push()在末尾加入一个元素
size()返回队列中元素的个数
queue 的基本操作举例如下：
queue 入队，如例：q.push(x); 将 x 接到队列的末端。
queue 出队，如例：q.pop(); 弹出队列的第一个元素，注意，并不会返回被弹出元素的值。
访问 queue 队首元素，如例：q.front()，即最早被压入队列的元素。
访问 queue 队尾元素，如例：q.back()，即最后被压入队列的元素。
判断 queue 队列空，如例：q.empty()，当队列空时，返回 true。
访问队列中的元素个数，如例：q.size()
#C++堆栈操作 #include <stack>
empty() 堆栈为空则返回真
pop() 移除栈顶元素
push() 在栈顶增加元素
size() 返回栈中元素数目
top() 返回栈顶元素
#dijistra
算法步骤如下：
G={V,E}
1. 初始时令 S={V0},T=V-S={其余顶点}，T 中顶点对应的距离值
若存在<V0,Vi>，d(V0,Vi)为<V0,Vi>弧上的权值
若不存在<V0,Vi>，d(V0,Vi)为∞
2. 从 T 中选取一个与 S 中顶点有关联边且权值最小的顶点 W，加入到 S 中

3. 对其余 T 中顶点的距离值进行修改：若加进 W 作中间顶点，从 V0 到 Vi 的距离值缩短，则修改此距离值（重复上述步骤 2、3，直到 S 中包含所有顶点，即 W=Vi 为止）

#qsort

```
void quicksort(int r[1001],int s,int e)
{
  int t = r[s];//哨兵，为开头的那个
  int f = s+1;
  int b = e;//f 为前向指针，从 s+1 开始，b 为反向指针，从 e 开始
  int m = 0;
  if(s>=e)return;//退出条件

  while(f<=b)
  {
     while(f<=b&&r[f]<=t) f++;//在前面找比哨兵大的元素
     while(f<=b&&r[b]>=t) b--;//在后面找比哨兵小的元素
     //交换这两个元素
     if(f<b){
         m = r[f];
         r[f] = r[b];
         r[b] = m;
         f++;    b--;
     }
  }
  //交换哨兵和 r[b],r[b]肯定要比哨兵小
  r[s] = r[b];
  r[b] = t;
  //排两边的
  quicksort(r,s,b-1);
  quicksort(r,b+1,e);
}
```