

**课程 实 验 报 告**

**课程名称： C语言程序设计课程实验**

**专业班级： 自实1901**

**学 号： U201916457**

**姓 名： 张皓然**

**指导教师：**

**报告日期： 2020年4月13日**

**人工智能与自动化学院**

# 实验1 常量字符串和一般字符串处理的数据类型选择有何差异？用实例分别说明（含单个字符串和多个字符串）

## 实验目的

区分常量字符串和一般字符串处理的数据类型选择的差异。

## 设计思路与代码实现

### 设计思路

分别处理常量字符串和一般字符串。

### 代码实现

对于常量字符串的处理：

1. #include<stdio.h>
2. **int** main()
3. {
4. **char**\* a = "hello";
5. **char**\* b[3] = {"hello", "HUST", "AIA"};
6. **int** i;
7. puts(a);
8. **for**(i = 0; i < 3; i++)
9. {
10. puts(b[i]);
11. }
12. **return** 0;
13. }

对于一般字符串的处理：

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **int** main()
5. {
6. **char** a[50];
7. **char** b[3][50];
8. **char**\* p;
9. **char**\* q[3];
10. **int** i;
11. gets(a);
12. **for**(i = 0;i < 3; i++)
13. {
14. gets(b[i]);
15. }
16. **if**((p = (**char** \*)malloc(strlen(a) + 1)) == NULL)
17. {
18. printf("\nmalloc failed");
19. }
20. strcpy(p, a);
21. puts(a);
22. free(p);
23. **for**(i = 0; i < 3; i++)
24. {
25. q[i] = (**char**\*)malloc(strlen(b[i])+1);
26. strcpy(q[i],b[i]);
27. }
28. **for**(i = 0;i < 3; i++)
29. {
30. puts(q[i]);
31. }
32. **return** 0;
33. }

## 测试集

this is the first part.

this is the first sentence.

this is the second sentence.

this is the third sentence.

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

this is the first part.

this is the first sentence.

this is the second sentence.

this is the third sentence.

## 实验分析

我们一般用指针处理单个常量字符串，用字符数组或是动态内存分配来处理单个一般字符串。对于多个字符串，用字符指针数组处理多个常量字符串，用多维字符数组或者动态内存分配来处理多个字符串。

在处理一个或多个常量字符串时，我们选择使用字符指针（字符指针数组）。首先系统开辟一块地址并把常量串放在地址中，然后再将指针指向这个地址。我们可以对指针重新赋值（指向其他地方），不过却无法改变常量串的内容。

在处理一个或多个一般字符串时，我们选择使用字符数组或是动态分配内存。字符数组的优点在于简单，缺点在于会浪费系统的内存（因为定义数组时，为保证数据不会溢出，我们总是把数组设置得很大）。而使用动态分配内存则可以很好的解决这个问题。

# 实验2 上机验证教材7.10，7.11, 7.12，7.13, 7.14

## 实验目的

上机验证教材上的例子

## 例7.10

编写一个函数，删除给定字符串中的数字字符。

### 代码

1. #include<stdio.h>
2. #include<string.h>
3. **char** \*delnum(**char**\*s);
4. **int** main()
5. {
6. **char** str[80];
7. printf("input string:\n");
8. gets(str);
9. puts(delnum(str));
10. }
11. **char**\* delnum(**char**\* s)
12. {
13. **int** i;
14. **char**\* temp=s;
15. **for**(i=0;s[i]!='\0';)
16. {
17. **if**(s[i]>='0'&&s[i]<='9')
18. {
19. strcpy(s,s+1);
20. }**else**
21. {
22. s++;
23. }
24. }
25. **return** temp;
26. }

### 过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 测试集

123abc…

a1b2c3…

abc123…

abc.123..

### 结果

abc…

abc…

abc…

abc…

### 分析

该例选择使用字符数组来处理单个的字符串，并设计了delnum函数来实现需求。delnum先用一个字符指针指向字符数组，程序逐位判断字符是否是数字类型的，如果是，利用strcpy函数，用s+1指向的字符串覆盖住s指向的字符串。如果不是，s++，使得字符指针向下移动一位。需要注意的是，在用strcpy函数覆盖住前一位后，依旧要对当位字符进行一次判断。

## 例7.11

删除字符串除前导\*以外的其他\*号。如\*\*\*\*A\*BC\*DEF\*\*\*\*转化为\*\*\*\*ABCDEF

### 代码

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. #include<conio.h>
5. **void** del(**char**\*s,**char**\*p);
6. **int** main()
7. {
8. **char** str[20],\*p;
9. printf("enter a string:\n");
10. gets(str);
11. **if**( (p=(**char**\*)malloc(strlen(str)+1) )==NULL)
12. {
13. printf("\n not enough memory to allocate buffer!");
14. exit(1);
15. }
16. del(str,p);
17. strcpy(str,p);
18. free(p);
19. printf("%s\n",str);
20. getch();
21. **return** 0;
22. }
23. **void** del(**char**\*s,**char**\*p)
24. {
25. **while**(\*s&&\*s=='\*')
26. {
27. \*p=\*s;
28. p++;
29. s++;
30. }
31. **while**(\*s)
32. {
33. **if**(\*s!='\*')
34. {
35. \*p=\*s;
36. p++;
37. }
38. s++;
39. }
40. \*p='\0';
41. }

### 过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 测试集

\*\*\*abc\*ab\*\*\*\*ab

### 结果

\*\*\*abcabab

### 分析

本例需要注意的地方如下：

第一，用动态内存分配来处理字符串。用stdlib.h中的库函数malloc和free来调用内存与释放内存。需要注意的是，如果malloc没能为程序分配到足够的内存，那么这个函数就会返回一个空指针。为了避免这种情况，我们一般会在使用malloc函数的同时，判断一下他是不是返回了一个空指针，如果是，就直接结束程序。

第二，exit函数。这里用到了exit函数。return是函数的退出，exit是进程的退出。exit(1)表示进程非正常结束，exit(0)表示进程正常结束。在用户自己定义的函数中，exit和return有明显的差别，但在main函数中，差别就不那么明显了。

第三，getch函数。在main函数的末尾，程序使用了getch函数。这个函数可以从键盘读入一个字符。实际的作用就是在程序实际完成后，留在用户界面，在任意输入一个字符后结束。此外，最后输入的这个字符不会显示在用户界面上。最后的最后，getch函数是在conio.h中的库函数。

## 例7.12

用字符指针数组处理多个字符串的排序问题

### 代码

1. #include<stdio.h>
2. #include<string.h>
3. **void** sortstr(**char**\*\*,**int** n);
4. **int** main()
5. {
6. **char**\* proname[]={"pascal","basic","cobol","prolog","lisp"};
7. **int** i;
8. sortstr(proname,5);
9. **for**(i=0;i<5;i++)
10. printf("%s\n",proname[i]);
11. **return** 0;
12. }
13. **void** sortstr(**char**\*\*v,**int** n)
14. {
15. **int** i,j;
16. **char**\*temp;
17. **for**(i=0;i<n-1;i++)
18. {
19. **for**(j=i+1;j<n;j++)
20. {
21. **if**(strcmp(v[i],v[j])>=0)
22. {
23. temp=v[i];
24. v[i]=v[j];
25. v[j]=temp;
26. }
27. }
28. }
29. }

### 过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 结果

basic

cobol

lisp

pascal

prolog

### 分析

该程序用字符指针数组对五个常量字符串进行的排序操作。其中排序过程是用strcmp函数加选择排序法实现的。需要注意的是，将字符指针数组传入函数需要两个参数，指针数组的地址和数组的行数。此外，从ISO C++开始，C语法实际上就不再支持将字符指针指向常量字符串了，未来如果在其他编译器上编写代码，需要注意这一点。

此外，本例中选择排序法实际上没有改变字符串的位置。起初指针数组内的五个指针依次指向了五个字符串，经过排序后，只是这五个指针的指向发生了变化。

## 例7.13

从键盘输入10个字符串，用选择法进行升序排序。

### 代码

1. #include<stdio.h>
2. #include<string.h>
3. **void** inpstr(**char**(\*p)[80],**int** n);
4. **void** sortstr(**char**(\*p)[80],**int** n);
5. **void** outstr(**char**(\*p)[80],**int** n);
6. **int** main()
7. {
8. **char** str[10][80];
9. inpstr(str,10);
10. sortstr(str,10);
11. outstr(str,10);
12. **return** 0;
13. }
14. **void** inpstr(**char**(\*p)[80],**int** n)
15. {
16. **int** i;
17. **for**(i=0;i<n;i++)
18. {
19. gets(p[i]);
20. }
21. }
22. **void** sortstr(**char**(\*p)[80],**int** n)
23. {
24. **int** i,j;
25. **char** temp[80];
26. **for**(i=0;i<n-1;i++)
27. {
28. **for**(j=i+1;j<n;j++)
29. {
30. **if**(strcmp(p[i],p[j])>0)
31. {
32. strcpy(temp,p[i]);
33. strcpy(p[i],p[j]);
34. strcpy(p[j],temp);
35. }
36. }
37. }
38. }
39. **void** outstr(**char**(\*p)[80],**int** n)
40. {
41. **int** i;
42. **for**(i=0;i<n;i++)
43. {
44. puts(p[i]);
45. }
46. }

### 过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 测试集

publisher

word

excel

access

frontpage

outlook

onenote

infopath

powerpoint

visio

### 结果

access

excel

frontpage

infopath

onenote

outlook

powerpoint

publisher

visio

word

### 分析

本例和上例都使用了选择排序法对字符串进行排序。不同的是，例7.12中用字符指针数组指向常量字符串，字符串的内容无法修改，于是以改变指针指向的方法实现了排序。而本例中用多维字符数组来处理字符串，数组内的字符串是可修改的，于是用strcpy函数修改数组各行的内容，从而实现了排序操作。

## 例7.14

### 代码

1. #include<stdio.h>
2. #include<string.h>
3. #include<stdlib.h>
4. **void** sortsrt(**char**\*\*str);
5. **int** main()
6. {
7. **int** i;
8. **char**\*str[10]={NULL};
9. **char** temp[100];
10. **for**(i=0;i<10;i++)
11. {
12. gets(temp);
13. str[i]=(**char**\*)malloc(strlen(temp)+1);
14. **if**(str[i]==NULL)
15. {
16. printf("not enough memory to allocate buffer.\n");
17. exit(1);
18. }
19. strcpy(str[i],temp);
20. }
21. sortstr(str,10);
22. **for**(i=0;i<n-1;i++)
23. {
24. puts(str[i]);
25. }
26. **for**(i=0;i<10;i++)
27. {
28. **if**(str[i]!=NULL)
29. {
30. free(str[i]);
31. }
32. }
33. **return** 0;
34. }
35. **void** sortstr(**char**\*\*str,**int** n)
36. {
37. **int** i,j;
38. **char**\* p;
39. **for**(i=0;i<n-1-i;j++)
40. {
41. **if**(strcmp(str[j],str[j+1])>=0)
42. {
43. p=str[j];
44. str[j]=str[j+1];
45. str[j+1]=p;
46. }
47. }
48. }

### 过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 分析

结合以上三个例子，我们对字符串排序进行一个小的总结。

1.用字符指针数组指向多个常量字符串。通过改变指针指向进行排序，因为常量字符串的内容不可修改。

2.用多维字符数组装入多个一般字符串。排序用strcpy更改各行的字符，因为数组名是地址常量，不能指向其他地方。

3.用动态内存分配来处理多个一般字符串。通过改变指针指向进行排序，因为动态内存分配出的存储空间刚刚够初始的字符串，虽然可以修改，但交换字符串的位置可能导致数据的溢出。

# 实验3 编写一个函数，统计字符串中子串出现的次数，如在字符串“10101000101”中出现子串“101”的个数为2；主函数完成字符串和子串的输入，调用所编函数得到子串出现的次数，并输出出现的次数。

## 实验目的

完成题目

## 设计思路与代码实现

### 设计思路

用strstr函数来搜索子串。搜索到字串后，改变字符指针的位置，并用strcpy覆盖掉子串，并记一次数。

### 代码实现

1. #include<stdio.h>
2. #include<string.h>
3. #include<stdlib.h>
4. **void** cnt(**char**\*,**char**\*);
5. **int** main()
6. {
7. **char** s1[80];
8. **char** s2[80];
9. **char**\*p1,\*p2;
10. printf("input the char:\n");
11. gets(s1);
12. printf("input the sub-char:\n");
13. gets(s2);
14. **if**( ( p1=(**char**\*)malloc(strlen(s1)+1) )==NULL)
15. {
16. printf("not enough memory to allocate buffer\n");
17. exit(1);
18. }
19. strcpy(p1,s1);
20. **if**( ( p2=(**char**\*)malloc(strlen(s2)+1) )==NULL)
21. {
22. printf("not enough memory to allocate buffer\n");
23. exit(1);
24. }
25. strcpy(p2,s2);
26. cnt(p1,p2);
27. **return** 0;
28. }
29. **void** cnt(**char**\*p,**char**\*subp)
30. {
31. **int** i=0;
32. **char**\* temp1=p;
33. **char**\* temp2=subp;
34. **while**(strstr(p,subp)!=NULL)
35. {
36. p=strstr(p,subp);
37. strcpy(p,p+strlen(subp));
38. i++;
39. }
40. printf("%d\n",i);
41. free(temp1);
42. free(temp2);
43. }

## 测试集

10101000101 101

zhanghaoranhaoranra ran

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

2

2

实验结果符合预期，程序设计成功。

# 实验4 编写一个函数，将字符串中的子串str1替换为str2，主函数完成字符串和两个子串的输入，调用所编函数得到新的字符串，并输出

## 实验目的

完成题目

## 设计思路与代码实现

### 设计思路

利用srtcpy和strcat函数来换成子字符串的替换。

### 代码实现

1. #include<stdio.h>
2. #include<string.h>
3. #include<stdlib.h>
4. **void** change(**char**\*,**char**\*,**char**\*);
5. **int** main()
6. {
7. **char** s1[100];
8. **char** s2[100];
9. **char** s3[100];
10. **char**\*p2,\*p3;
11. printf("input the char:\n");
12. gets(s1);
13. printf("input the sub-char:\n");
14. gets(s2);
15. printf("input new sub-char:\n");
16. gets(s3);
17. **if**( ( p2=(**char**\*)malloc(strlen(s2)+1) )==NULL)
18. {
19. printf("not enough memory to allocate buffer\n");
20. exit(1);
21. }
22. strcpy(p2,s2);
23. **if**( ( p3=(**char**\*)malloc(strlen(s3)+1) )==NULL)
24. {
25. printf("not enough memory to allocate buffer\n");
26. exit(1);
27. }
28. strcpy(p3,s3);
29. change(s1,p2,p3);
30. **return** 0;
31. }
32. **void** change(**char**\*p,**char**\*subp,**char**\*newsubchar)
33. {
34. **char**\* temp1=p;
35. **char**\* temp2=subp;
36. **char**\* temp3=newsubchar;
37. **char** a[100];
38. **while**(strstr(p,subp)!=NULL)
39. {
40. p=strstr(p,subp);
41. strcpy(a,p+strlen(subp));
42. strcpy(p,newsubchar);
43. strcat(p,a);
44. }
45. puts(temp1);
46. free(temp1);
47. free(temp2);
48. free(temp3);
49. }

## 测试集

1234567 234 6789

1234567 234 67

1234567 234 789

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

16789567

167567

1789567

结果与预期相符，程序设计成功。

# 实验5 P200教材习题7中的“四、编程题”的1、2、3、7、9、11、12

## 实验目的

完成题目

## 题目1

编写一个函数，由实参传递一个字符串，统计此字符串中字母，数字，空格和其他字符的个数，在主函数中输入字符串及输出上述统计的结果。

### 设计思路

利用ASCII码中不同种类字符的排列规律进行分类统计。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** f(**char**\*,**int**[]);
5. **int** main()
6. {
7. **char** a[100];
8. **char**\* p;
9. **int** num[4]={0};
10. **int** i;
11. gets(a);
12. **if**((p=(**char**\*)malloc(strlen(a)+1))==NULL)  //动态内存分配
13. {
14. printf("not enough memory to allocate buffer\n");
15. exit(1);
16. }
17. strcpy(p,a);  //将字符串放进刚刚分配的内存中
18. f(p,num);
19. **for**(i=0;i<4;i++)
20. {
21. printf("%d\t",num[i]);
22. }
23. **return** 0;
24. }
25. **void** f(**char**\*p,**int** a[])
26. {
27. **int** cnt=0;
28. **while**(p[cnt]!='\0')
29. {
30. **if**(p[cnt]>='A'&&p[cnt]<='z')
31. a[0]++;
32. **else** **if**(p[cnt]>='0'&&p[cnt]<='9')
33. a[1]++;
34. **else** **if**(p[cnt]==' ')
35. a[2]++;
36. **else**
37. a[3]++;
38. cnt++;
39. }
40. }

### 测试集

abc123…

abc\*\* \*\* 33

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

3 3 0 3

3 2 3 4

实验结果符合预期。

## 题目2

编写一个函数，在字符串中的指定位置插入一个子串，如在字符串”abcghi”中第三个字符后插入子串”def”为”abcdefghi”，如插入位置不合法，原字符串不作任何处理；主函数完成字符串，插入位置，子串的输入，调用所编函数得到插入后的字符串，并输出。

### 设计思路

利用strcpy和strcat函数来实现子串的插入。另外，在scanf后面使用gets函数，会使得gets函数得到scanf输入结束时的那个回车，我额外添加了一个字符变量来解决这件事情。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** f(**char**\*,**char**\*,**int**);
5. **int** main()
6. {
7. **char** a[100],b[100];
8. **char** enter[1];
9. **int** position;
10. printf("input char:\n");
11. gets(a);
12. printf("input position number:\n");
13. scanf("%d",&position);
14. **if**(position<1||position>strlen(a))  //如果插入位置不合法
15. {
16. printf("wrong!\n");
17. exit(1);
18. }
19. gets(enter);//\*\*\*读入scanf的那个回车\*\*\*
20. printf("input sub-char:\n"); //子字符串
21. gets(b);
22. f(a,b,position);
23. **return** 0;
24. }
25. **void** f(**char**\*a,**char**\*b,**int** p)
26. {
27. **char** temp[100];
28. strcpy(temp,a+p);  //复制后半部分的字符串
29. a[p]='\0';   //截断字符串，前半部分形成独立的字符串
30. strcat(a,b);   //拼接前半部分后子串
31. strcat(a,temp);   //子串再和后半部分拼接
32. puts(a);
33. }

### 测试集

abcghi 3 def

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

abcdefghi

实验结果符合预期。

## 题目3

判断字符串是否是回文

### 设计思路

将字符串顺序跟逆序比较。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **int** fun(**char**\*);  //返回0或1
5. **int** main()
6. {
7. **char** a[100];
8. **int** sign;
9. gets(a);
10. sign=fun(a);
11. **if**(sign)
12. printf("YES\n");
13. **else**
14. printf("NO\n");
15. **return** 0;
16. }
17. **int** fun(**char**\* a)
18. {
19. **char**\* p;
20. **int** i=0,j=0;
21. **int** sign;
22. **if**(( p=(**char**\*)malloc(strlen(a)+1) )==NULL)  //动态内存分配，
23. {
24. printf("not enough memory to allocate buffer\n");
25. exit(1);
26. }
27. **while**(a[i]!='\0')  //找到a的末尾
28. {
29. i++;
30. }
31. i--;
32. **while**(i>=0)  //将a逆序导入p
33. {
34. p[j++]=a[i--];
35. }
36. p[j]='\0';  //为p结尾
37. **if**(strcmp(a,p)==0)  //比较
38. sign=1;
39. **else**
40. sign=0;
41. **return** sign;
42. }

### 测试集

level

hanna

abcba

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

YES

NO

YES

实验结果与预期相符。

## 题目4

输入八进制的字符串，转化成十进制字符串。

### 设计思路

通过atoi等函数完成操作。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** change(**char**\*);
5. **int** main()
6. {
7. **char** a[50];
8. gets(a);
9. change(a);
10. **return** 0;
11. }
12. **void** change(**char**\* a)
13. {
14. **int** sum = 0;  //注意sum要初始为0
15. **char**\* temp = a;  //后期a的指向改变了，这里预留一个temp
16. **while** (\*a != '\0')
17. {
18. sum = sum \* 8 + \*a-48;  //利用ASCII码，将八进制字符转为十进制数字
19. a++;
20. }
21. itoa(sum, temp, 10);  //a的指向早已改变，我们用temp
22. puts(temp);
23. }

### 测试集

1732

2327

6360

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

986

1239

3312

实验结果与预期相符。

## 题目7

输入10个字符串，排序输出。排序的原则由键盘输入的数决定，若为0，讲字符串俺整数值大小从小到大排序，否则按字典顺序排序。要求，输入，输出，排序分别用函数实现，主函数只是调用这些函数。

### 设计思路

设计两个排序函数，依rule来决定使用哪个排序函数。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** getsc(**char**(\*)[80],**int**,**int**\*);  //输入函数
5. **void** putsc(**char**(\*)[80],**int**);   //输出函数
6. **void** sorts(**char**(\*)[80],**int**,**int**);   //主排序函数，内进行判断
7. **void** sorts1(**char**(\*)[80],**int**);   //数值大小排序函数
8. **void** sorts2(**char**(\*)[80],**int**);   //字典排序函数
9. **int** main()
10. {
11. **char** a[10][80];
12. **int** n;  //排序规则
13. getsc(a,10,&n);
14. sorts(a,10,n);
15. putsc(a,10);
16. **return** 0;
17. }
18. **void** getsc(**char**(\*a)[80],**int** line,**int**\* pn)
19. {
20. **int** i;
21. **for**(i=0;i<line;i++)
22. gets(a[i]);
23. printf("input the rule\n");
24. scanf("%d",pn);
25. }
26. **void** putsc(**char**(\*a)[80],**int** line)
27. {
28. **int** i;
29. **for**(i=0;i<line;i++)
30. puts(a[i]);
31. }
32. **void** sorts(**char**(\*a)[80],**int** line,**int** n)
33. {
34. **if**(n==0)  //判断规则，然后执行对应的排序函数
35. {
36. sorts1(a,line);
37. }**else**
38. {
39. sorts2(a,line);
40. }
41. }
42. **void** sorts1(**char**(\*a)[80],**int** line)
43. {
44. **char** temp[80];  //用于交换位置
45. **int** t;  //用于交换位置
46. **int** i,j;
47. **long** b[10];  //将字符串转为数字
48. **for**(i=0;i<line;i++)  //字符串变为数字
49. {
50. b[i]=atoi(a[i]);
51. }
53. **for**(i=0;i<line-1;i++)  //选择排序法
54. {
55. **for**(j=i+1;j<line;j++)
56. {
57. **if**(b[i]>b[j])
58. {
59. t=b[i];  //数值的交换
60. b[i]=b[j];
61. b[j]=t;
63. strcpy(temp,a[i]);  //字符串的交换
64. strcpy(a[i],a[j]);
65. strcpy(a[j],temp);
66. }
67. }
68. }
69. }
70. **void** sorts2(**char**(\*a)[80],**int** line)
71. {
72. **int** i,j;
73. **char** temp[80];
74. **for**(i=0;i<line-1;i++)  //选择排序法
75. {
76. **for**(j=i+1;j<line;j++)
77. {
78. **if**(strcmp(a[i],a[j])>0)
79. {
80. strcpy(temp,a[i]);
81. strcpy(a[i],a[j]);
82. strcpy(a[j],temp);
83. }
84. }
85. }
86. }

### 测试集

1 2 3 4 5 6 7 8 9 10 rule:0

1 2 3 4 5 6 7 8 9 10 rule:1

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

1

2

3

4

5

6

7

8

9

10

1

10

2

3

4

5

6

7

8

9

实验结果与预期相符。

## 题目9

输入一行英文，调用函数求该行英文中最大（字典排序）的那个单词。

### 设计思路

利用strcmp函数进行字典排序的比较。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** finds(**char**\*);
5. **int** main()
6. {
7. **char** a[100];
8. gets(a);
9. finds(a);
10. **return** 0;
11. }
12. **void** finds(**char**\*word)
13. {
14. **char**\* pword[20]={NULL};  //指向各个单词
15. **char**\* maxs=NULL;
16. **int** i=0;
17. pword[i]=word;
19. **for**(i=1;;i++)//定位每一个单词
20. {
21. **while**(\*word!=' '&&\*word!='\0')  //找到空格
22. word++;
23. **if**(\*word=='\0')  //抵达’0’,循环结束
24. **break**;
25. \*word='\0';  //空格改写为‘0’，分隔开各个单词
26. pword[i]=word+1;
27. word++;
28. }
29. maxs=pword[0];
30. **for** (i = 0; (pword[i] != NULL) && (i < 20); i++)//找最大的单词
31. {
32. **if**( strcmp(maxs,pword[i])<0 )
33. maxs=pword[i];
34. }
35. puts(maxs);
36. }

### 测试集

xiaoming walks to the zoo

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

zoo

实验结果与预期相符。

## 题目11

字符串含有数字字符与非数字字符。输出其中最大的数字字符串。

### 设计思路

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** number(**char**\*);
5. **int** main()
6. {
7. **char** a[80];
8. gets(a);
9. number(a);
10. puts(a);
11. **return** 0;
12. }
14. **void** number(**char**\*a)
15. {
16. **char**\*tempp=a;  //预留a的地址
17. **char** temp[10][80]={"\0","\0","\0","\0","\0","\0","\0","\0","\0","\0"};
19. **char**\* aend=NULL;  //每串数字的末地址
20. **char**\* maxs=NULL;
21. **char** t;
22. **int** i=0;
24. **for**(i=0;\*a!='\0';i++)
25. {
26. //\*\*\*\*\*定位数字子串\*\*\*\*\*a为首地址，aend为尾地址
27. **while**(\*a<'0'||\*a>'9')  //寻找数字
28. a++;
29. aend=a;  //开始定位数字的末尾
30. **while**( (\*aend>='0' && \*aend<='9') ||(\*aend=='.'&&( \*(aend+1)>='0'&& \*(aend+1)<='9' ) ) )
31. aend++;
32. t=\*aend;  //\*\*\*\*
33. \*aend='\0';
34. **if**(\*(a-1)=='-')  //判断是否是负数
35. {
36. a--;
37. }
38. strcpy(temp[i],a);
39. \*aend=t;
40. a=aend;
41. }

44. //统一用浮点数比较
45. i=1;
46. maxs=temp[0];
47. **while**( (temp[i][0]!='\0') && i<10 )
48. {
49. **if**(atof(temp[i])>atof(maxs))
50. maxs=temp[i];
51. i++;
52. }
53. a=tempp;
54. strcpy(a,maxs);
55. }

### 测试集

a123b345.6x876.1y76t

12.4zb-999.9abc120d0

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

876.1

120

实验结果与预期相符。

## 题目12

输入一行包含若干单词的字符串，单词之间用空格分开。按单词长短从小到大的次序排序后形成新的字符串输出。

### 设计思路

利用strlen函数。

### 代码实现

1. #include<stdio.h>
2. #include<stdlib.h>
3. #include<string.h>
4. **void** points(**char**\*, **char**\* []);
5. **void** sorts(**char**\* [], **int**);
6. **void** putsc(**char**\* [], **int**);
7. **int** main()
8. {
9. **char** word[100];
10. **char**\* pword[10] = { NULL };
11. gets(word);
12. points(word, pword);
13. sorts(pword, 10);
14. putsc(pword, 10);
15. **return** 0;
16. }
17. **void** points(**char**\* word, **char**\* pword[])
18. {
19. **int** i = 0;
20. //\*\*\*\*\*标记每一个单词\*\*\*\*\*
21. pword[i] = word;
22. **for** (i = 1;; i++)
23. {
24. **while** (\*word != ' ' && \*word != '\0')
25. word++;
26. **if** (\*word == '\0')
27. **break**;
28. \*word = '\0';  //将每个单词分隔开
29. pword[i] = word + 1;
30. word++;
31. }
32. }
33. **void** sorts(**char**\* pword[], **int** n)
34. {
35. **char**\* temp;
36. **int** i, j;
37. **for** (i = 0; (pword[i+1] != NULL) && (i < n); i++)  //选择排序法
38. {
39. **for** (j = i + 1; (pword[j] != NULL)&&(j<n); j++)
40. {
41. **if** (strlen(pword[i]) > strlen(pword[j]))
42. {
43. temp = pword[j];
44. pword[j] = pword[i];
45. pword[i] = temp;
46. }
47. }
48. }
49. }
50. **void** putsc(**char**\* pword[], **int** n)
51. {
52. **int** i;
53. **for** (i = 0; (pword[i] != NULL) && (i < n); i++)
54. puts(pword[i]);
55. }

### 测试集

abcdefg abcdef abcde abcd abc ab a ss dhw swe

abssw swbf

### 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

### 实验结果

a

ab

ss

abc

dhw

swe

abcd

abcde

abcdef

abcdefg

swbf

abssw

实验结果与预期相符。

附录

## 开发环境

本次实验中使用的环境配置如下：

（1）操作系统版本：Arch Linux x86\_64

（2）编译器及其版本：GCC version 8.1.1

（3）自动编译工具：CMake version 3.11.4

（4）编程环境：Borland C++ 3.1