文字列処理

０．目次

１．文字と文字定数

２．文字列、文字列定数、文字列配列

３．文字列の長さ

４．文字列の分離

５．文字列の連結

６．文字列関数

６．１　文字列の長さを求める関数
６．２　文字列を複写する関数
６．３　文字列を連結する関数
６．４　文字列を比較する関数
６．５　文字列から数値への変換

７　課題

７．１　課題１
７．２　課題２
７．３　課題３
７．４　課題４

１．文字と文字定数

文字の宣言	char ch
文字の入力	scanf("%c",&ch)
文字とコード	文字はそのまま８ビットの数値（文字コード）とみなされる。
文字定数	文字定数は'で囲む。

```
1  /* << c111.c >> */
2  #include <stdio.h>
3  main() {
4    int i;
5    char ch; /* 文字の宣言 */
6    scanf("%c",&ch); /* 文字の入力 */
7    printf("文字       : %c ¥n",ch);
8    printf("文字コード : %d ¥n",ch);
9    ch = '0'; /* 文字定数 */
10   printf("文字       : %c ¥n",ch);
11   printf("文字コード : %d ¥n",ch);
12 }
```

実行結果

```
% cc c111.c
% a.out
1
文字       : 1
文字コード : 49
文字       : 0
文字コード : 48
```

２．文字列、文字列定数、文字列配列

文字列の宣言	**char x[9]** 文字列は配列に格納される。
文字列の入力	**scanf("%s",x)** 読み込んだ文字は、配列要素のx[0],x[1],x[2]の順に格納され、最後に文字コード0の文字('¥0'で表す)が格納される。 したがって、配列の大きさは、読み込む文字列の長さ+1以上でなければならない。

●文字列

```
 1  /* << c211.c >> */
 2  #include <stdio.h>
 3  main() {
 4    int i;
 5    char x[9];      /* 文字列の宣言 */
 6    scanf("%s",x); /* 文字列の入力 */
 7    /* 文字列の出力 */
 8    printf("文字列：|%s| ¥n",x);    /* 文字列の長さ分の桁に出力*/
 9    printf("文字列：|%8s| ¥n",x);   /* ８桁の中に右寄せで出力  */
10    printf("文字列：|%-8s| ¥n",x); /* ８桁の中に左寄せで出力  */
11    /* １文字ずつの処理 */
12    for( i=0; i<9; i++ ) {
13      printf("x[%d]  ",i);    /* 配列要素 */
14      printf("|%d|  ",x[i]); /* 10進数   */
15      printf("|%x|  ",x[i]); /* 16進数   */
16      printf("|%c|  ",x[i]); /* 文字     */
17      printf("¥n");
18    }
19  }
```

実行結果

```
% cc c211.c
% a.out
abcABC
文字列：|abcABC|
文字列：|  abcABC|
文字列：|abcABC  |
x[0]  |97|  |61|  |a|
x[1]  |98|  |62|  |b|
x[2]  |99|  |63|  |c|
x[3]  |65|  |41|  |A|
x[4]  |66|  |42|  |B|
x[5]  |67|  |43|  |C|
x[6]  |0|  |0|  ||
x[7]  |0|  |0|  ||
x[8]  |0|  |0|  ||
```

配列 x

0	1	2	3	4	5	6	7	8
a	b	c	A	B	C	¥0	¥0	¥0

●文字列定数

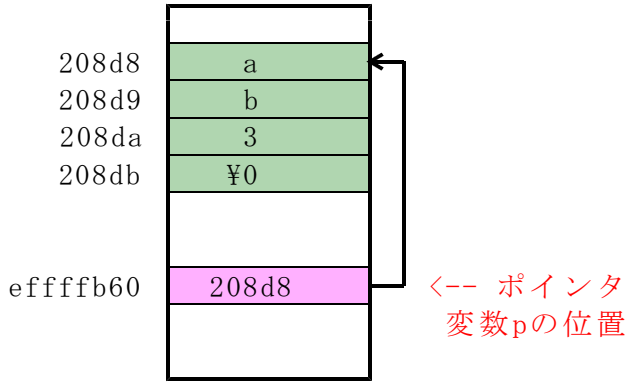
文字列定数の宣言	**char *p** 文字列定数は、アドレスで示されるのでポインタ変数（アドレスを格納するための変数）を使う。 ポインタ変数で指されたアドレスから順に文字が格納され、最後に文字コード0の文字（'¥0'で表す）が格納される。
文字列定数	**p = "abc123"** 文字列定数は"で囲む。

```
 1  /* << c221.c >> */
 2  #include <stdio.h>
 3  main() {
 4    int i;
 5    char *p; /* ポインタ変数の宣言 */
 6    p = "ab3"; /* 文字列ab3を記憶場所に保存しその先頭の
 7                  アドレスをポインタ変数pに格納する。*/
 8    /* 文字列定数の出力 */
 9    printf("文字列定数 : %s ¥n",p);
10    printf("p+0:%x  *(p+0):|%c| ¥n",p+0,*(p+0));
11    printf("p+1:%x  *(p+1):|%c| ¥n",p+1,*(p+1));
12    printf("p+2:%x  *(p+2):|%c| ¥n",p+2,*(p+2));
13    printf("p+3:%x  *(p+3):|%c| ¥n",p+3,*(p+3));
14    printf("&p:%x ¥n",&p);
15  }
```

実行結果

```
% cc c221.c
% a.out
文字列定数 : ab3
p+0:208d8  *(p+0):|a|
p+1:208d9  *(p+1):|b|
p+2:208da  *(p+2):|3|
p+3:208db  *(p+3):||
&p:effffb60
```

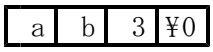
メモリ配置



処理手順

①**char *p** でポインタ変数pが確保される。

②**"ab3"**で文字列ab3が記憶場所に保存される。



③**p = "ab3"**で文字列先頭のアドレスがポインタ変数pに格納される。

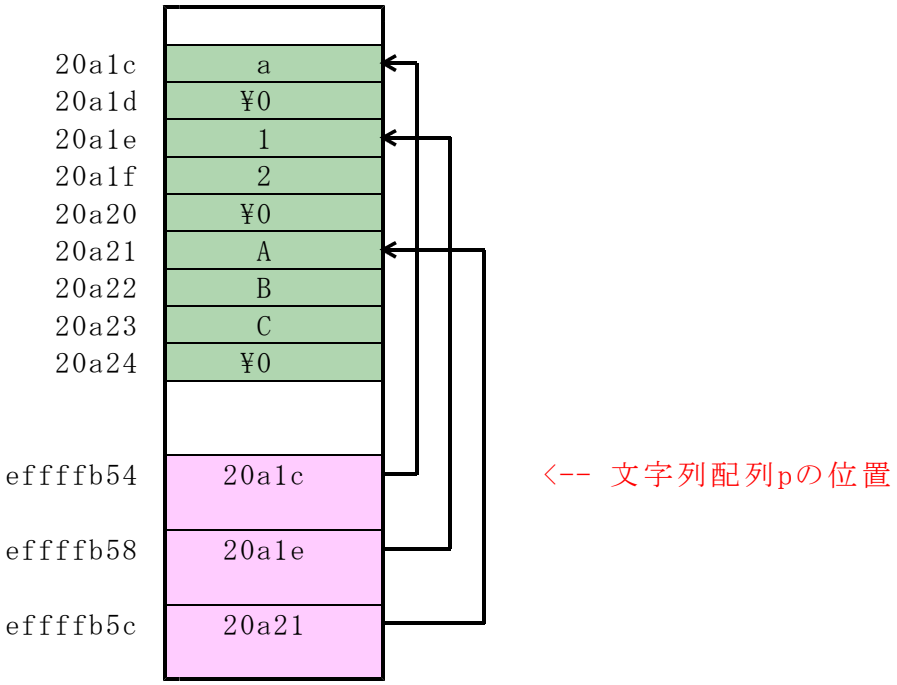
●文字列配列

```
1  /* << c231.c >> */
2  #include <stdio.h>
3  main() {
4    int i,j;
5    char *p[3]; /* 文字列配列の宣言 */
6    /* 文字型配列への代入 */
7    p[0] = "a"; /* 文字列aを記憶場所に保存しその先頭の
8                   アドレスをp[0]に格納する。*/
9    p[1] = "12";
10   p[2] = "ABC";
11   /* 文字列配列要素 */
12   for( j=0; j<=2; j++ ) {
13     i = 0;
14     while( 1 ) {
15       printf("p[%d]+%d:%x  ",j,i,p[j]+i); /* アドレス */
16       printf("*(p[%d]+%d):|%c| ¥n",j,i,*(p[j]+i)); /* 文字 */
17       if( *(p[j]+i) == '¥0' ) { break; }
18       i++;
19     }
20   }
21   printf("p:%x¥n",p);
22   printf("&p:%x¥n",&p);
23   for( j=0; j<=2; j++ ) { printf("&p[%d]:%x¥n",j,&p[j]); }
24 }
```

実行結果

```
% cc c231.c
% a.out
p[0]+0:20a1c  *(p[0]+0):|a|
p[0]+1:20a1d  *(p[0]+1):||
p[1]+0:20a1e  *(p[1]+0):|1|
p[1]+1:20a1f  *(p[1]+1):|2|
p[1]+2:20a20  *(p[1]+2):||
p[2]+0:20a21  *(p[2]+0):|A|
p[2]+1:20a22  *(p[2]+1):|B|
p[2]+2:20a23  *(p[2]+2):|C|
p[2]+3:20a24  *(p[2]+3):||
p:effffb54
&p:effffb54
&p[0]:effffb54
&p[1]:effffb58
&p[2]:effffb5c
```

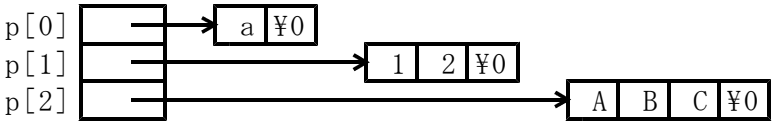
メモリ配置



処理手順

①**char *p[3]**によって、ポインタ変数を要素とする配列pが確保される。
要素数は３個。

②**p[0] = "a"** で文字列aが記憶場所に保存され、先頭のアドレスがp[0]に格納される。p[1],p[2]も同様。



３．文字列の長さ

```
 1  /* << c311.c >> */
 2  #include <stdio.h>
 3  main() {
 4    int len;      /* 文字列の長さ */
 5    char in[81]; /* 入力用配列の宣言 */
 6    scanf("%s",in);
 7    printf("文字列：%s ¥n",in);
 8    /* 文字コード0が現れるまで調べる。*/
 9    len = 0;
10    while( in[len] != '¥0' ) { len++; }
11    printf("文字列の長さ＝%d ¥n",len);
12  }
```

実行結果

```
% cc c311.c
% a.out
abcde
文字列：abcde
文字列の長さ＝5
```

４．文字列の分離

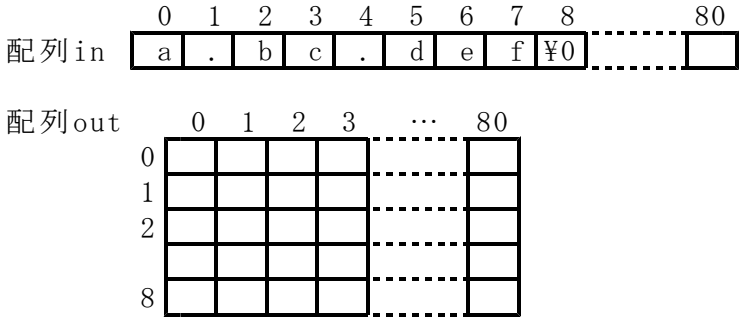
入力文字列をピリオドで分離する。処理前と処理後で入力文字列は変わらない。

```
1  /* << c411.c >> */
2  #include <stdio.h>
3  main() {
4    int i,j,k,
5        len,          /* 入力した文字列の長さ */
6        m;            /* 分離した文字列の個数 */
7    char in[81],      /* 入力用配列の宣言 */
8         out[9][81], /* 出力用配列の宣言 */
9         del;         /* 分離文字 */
10   /* 文字列の入力 */
11   scanf("%s",in);
12   printf("処理前文字列(in) : %s ¥n",in);
13   del = '.';
14   printf("分離文字 : %c ¥n",del);
15   /* 分離 */
16   len = strlen(in); j = 0; m = 0;
17   for( i=0; i<len; i++ ) {
18     if( in[i] != del ) {
19       out[m][j] = in[i]; j++;
20     } else {
21       out[m][j] = '¥0'; j = 0; m++;
22     }
23   }
24   if( in[len-1] != del ) { out[m][j] = '¥0'; m++; }
25   for( k=0; k<m; k++ ) {
26     printf("分離された文字列(%d) : %s ¥n",k,out[k]);
27   }
28   printf("処理後文字列(in) : %s ¥n",in);
29 }
```

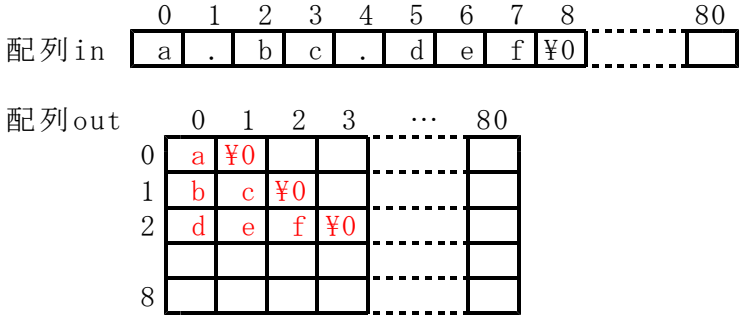
実行結果

```
% cc c411.c
% a.out
a.bc
処理前文字列(in) : a.bc
分離文字 : .
分離された文字列(0) : a
分離された文字列(1) : bc
処理後文字列(in) : a.bc
% a.out
a.bc.
処理前文字列(in) : a.bc.
分離文字 : .
分離された文字列(0) : a
分離された文字列(1) : bc
処理後文字列(in) : a.bc.
```

処理前



処理後



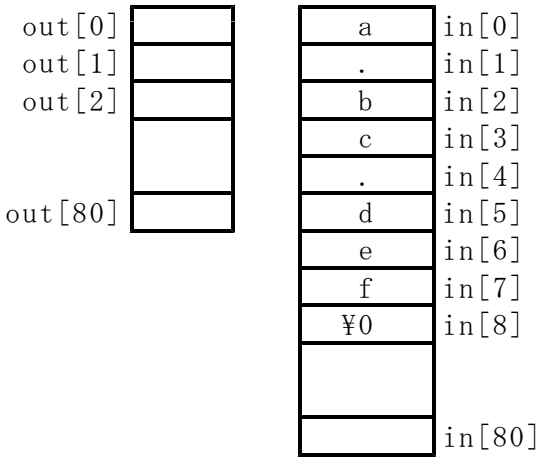
入力文字列をピリオドで分離する。処理後、入力文字列は変わる。

```
/* << c421.c >> */
#include <stdio.h>
main() {
  int i,k,
      len,        /* 文字列の長さ */
      m;          /* 分離した文字列の個数 */
  char in[81],    /* 入力用配列の宣言 */
       *out[81], /* 出力用配列の宣言 */
       del;       /* 分離文字 */
  scanf("%s",in); /* 文字列の入力 */
  printf("処理前文字列(in) : %s ¥n",in);
  del = '.';
  printf("分離文字 : %c ¥n",del);
  /* 分離 */
  len = strlen(in);
  m = 0; out[0] = in;
  for( i=0; i<len; i++ ) {
    if( in[i] == del ) {
      in[i] = '¥0'; m++; out[m] = in+i+1;
    }
  }
  if( in[len-1] != '¥0' ) { in[len] = '¥0'; m++; }
  for( k=0; k<m; k++ ) {
    printf("分離された文字列(%d) : %s ¥n",k,out[k]);
  }
  printf("処理後文字列(in) : %s ¥n",in);
}
```

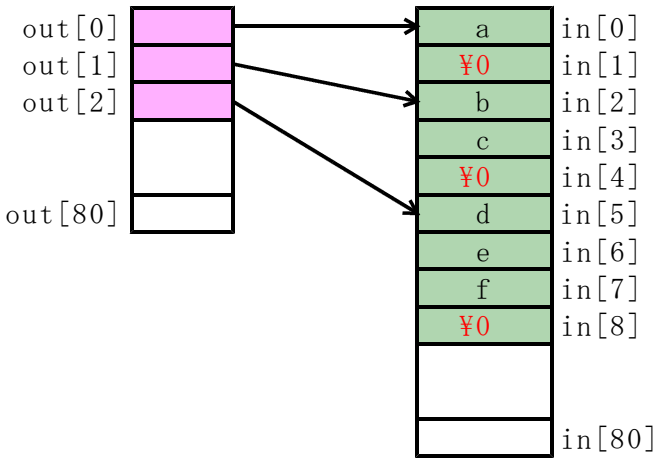
実行結果

```
% cc c421.c
% a.out
a.bc
処理前文字列(in) : a.bc
分離文字 : .
分離された文字列(0) : a
分離された文字列(1) : bc
処理後文字列(in) : a
% a.out
a.bc.
処理前文字列(in) : a.bc.
分離文字 : .
分離された文字列(0) : a
分離された文字列(1) : bc
処理後文字列(in) : a
```

処理前



処理後



５．文字列の連結

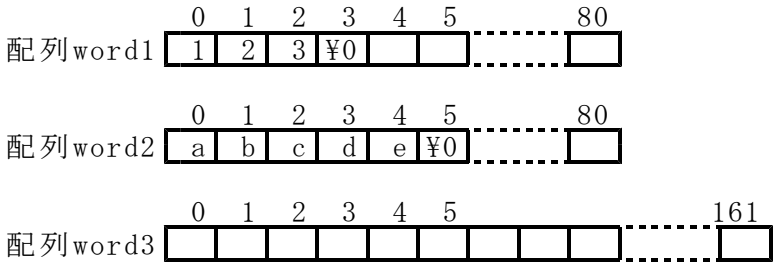
読み込んだ文字列１と文字列２を結合し文字列３を作る。
たとえば、文字列１＝abc,文字列２＝123ならば、文字列３＝abc123。

```
 1  /* << c511.c >> */
 2  #include <stdio.h>
 3  main() {
 4    int i,j;
 5    char word1[81],word2[81], /* 入力用配列の宣言 */
 6         word3[162];          /* 出力用配列の宣言 */
 7    /* 文字列の入力。*/
 8    scanf("%s",word1); scanf("%s",word2);
 9    printf("文字列１：%s ¥n",word1);
10    printf("文字列２：%s ¥n",word2);
11    i = 0; /* 入力用配列word1,word2の位置。*/
12    j = 0; /* 出力用配列word3の位置。*/
13    /* 連結 */
14    while( word1[i] != '¥0' ) { word3[j] = word1[i]; j++; i++; }
15    i = 0;
16    while( word2[i] != '¥0' ) { word3[j] = word2[i]; j++; i++; }
17    word3[j] = '¥0';
18    printf("文字列３：%s ¥n",word3);
19  }
```

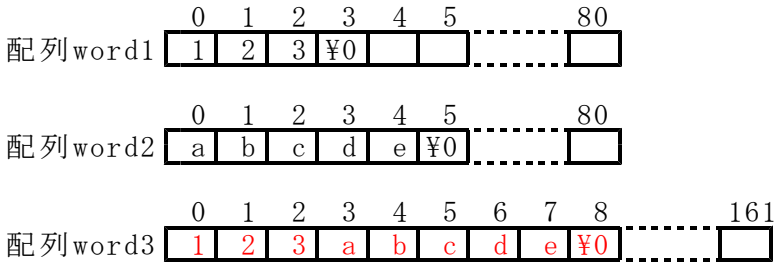
実行結果

```
% cc c511.c
% a.out
123
abcde
文字列１：123
文字列２：abcde
文字列３：123abcde
```

処理前



処理後



６．文字列関数

文字列を処理する便利な関数がいくつか用意されている。

６．１　文字列の長さを求める関数

関数	strlen
書き方	strlen(文字列);
機能	文字列の長さを返す。

```
1  /* << c611.c >> */
2  /* strlen関数 */
3  #include <stdio.h>
4  #include <strings.h> /* 文字列関数を使う場合必要。*/
5  main() {
6    char *str1;
7    str1 = "abc";
8    printf("str1:%s  str1の長さ:%d ¥n",str1,strlen(str1));
9  }
```

実行結果

```
% cc c611.c
% a.out
str1:abc  str1の長さ:3
```

６．２　文字列を複写する関数

関数	strcpy
書き方	strcpy(複写先文字列,複写元文字列);
機能	複写元文字列を複写先文字列に複写する。

```
1  /* << c621.c >> */
2  /* strcpy関数 */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    char str1[99],str2[99];
7    strcpy(str1,"abc");
8    strcpy(str2,"123");
9    printf("str1:%s  str2:%s ¥n",str1,str2);
10 }
```

実行結果

```
% cc c621.c
% a.out
str1:abc  str2:123
```

関数	strncpy
書き方	strncpy(複写先文字列,複写元文字列,複写文字数ｎ);
機能	複写元文字列の最初のｎ文字を複写先文字列に複写する。

```
1  /* << c622.c >> */
2  /* strncpy関数 */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    char str1[99],str2[99];
7    strncpy(str1,"abcde",2); str1[2] = '¥0';
8    strncpy(str2,"12345",4); str2[4] = '¥0';
9    printf("str1:%s  str2:%s ¥n",str1,str2);
10 }
```

実行結果

```
% cc c622.c
% a.out
str1:ab  str2:1234
```

```
1  /* << c623.c >> */
2  /* strncpy関数の応用（右からｎ文字分取り出す） */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    int n;
7    char str1[99],str2[99],str3[99];
8    strcpy(str1,"abcdefgh");
9    n = 5;
10   strncpy(str2,str1+(strlen(str1)-n),n);
11   str2[n] = '¥0';
12   printf("str1:%s  str2:%s ¥n",str1,str2);
13 }
```

実行結果

```
% cc c623.c
% a.out
str1:abcdefgh  str2:defgh
```

```
1  /* << c624.c >> */
2  /* strncpy関数の応用（ｋ番目の文字からｎ文字分取り出す） */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    int k,n;
7    char str1[99],str2[99];
8    strcpy(str1,"abcdefgh");
9    k = 2; n = 3;
10   strncpy(str2,str1+k,n);
11   printf("str1:%s  str2:%s ¥n",str1,str2);
12 }
```

実行結果

```
% cc c624.c
% a.out
str1:abcdefgh  str2:cde
```

６．３　文字列を連結する関数

関数	strcat
書き方	strcat(連結先文字列,連結元文字列);
機能	連結先文字列の末尾に連結元文字列を連結する。

```
 1 /* << c631.c >> */
 2 /* strcat関数 */
 3 #include <stdio.h>
 4 #include <strings.h>
 5 main() {
 6   char str1[99],str2[99];
 7   strcpy(str1,"abc");
 8   strcpy(str2,"123");
 9   printf("連結前   str1:%s   str2:%s ¥n",str1,str2);
10   strcat(str1,str2);
11   printf("連結後   str1:%s   str2:%s ¥n",str1,str2);
12 }
```

実行結果

```
% cc c631.c
% a.out
連結前   str1:abc   str2:123
連結後   str1:abc123   str2:123
```

関数	strncat
書き方	strncat(連結先文字列,連結元文字列,長さｎ);
機能	連結先文字列の末尾に連結元文字列の最初のｎ文字を連結する。

```
 1 /* << c632.c >> */
 2 /* strncat関数 */
 3 #include <stdio.h>
 4 #include <strings.h>
 5 main() {
 6   char str1[99],str2[99];
 7   strcpy(str1,"abc");
 8   strcpy(str2,"123");
 9   printf("連結前   str1:%s   str2:%s ¥n",str1,str2);
10   strncat(str1,str2,2);
11   printf("連結後   str1:%s   str2:%s ¥n",str1,str2);
12 }
```

実行結果

```
% cc c632.c
% a.out
連結前   str1:abc   str2:123
連結後   str1:abc12   str2:123
```

６．４　文字列を比較する関数

```
 1  /* << c641.c >> */
 2  /* strcmp関数 */
 3  #include <stdio.h>
 4  #include <strings.h>
 5  main() {
 6    char str1[99],str2[99],str3[99];
 7    strcpy(str1,"abc");
 8    strcpy(str2,"123");
 9    if( strcmp(str1,str2) > 0 ) {
10      printf("文字列：%s は文字列：%s より辞書式順序で後¥n",
11              str1,str2);
12    }
13    strcpy(str1,"abc");
14    strcpy(str2,"abc");
15    if( strcmp(str1,str2) == 0 ) {
16      printf("文字列：%s は文字列：%s と同じ¥n",str1,str2);
17    }
18    strcpy(str1,"ABC");
19    strcpy(str2,"abc");
20    if( strcmp(str1,str2) < 0 ) {
21      printf("文字列：%s は文字列：%s より辞書式順序で前¥n",
22              str1,str2);
23    }
24  }
```

実行結果

```
% cc c641.c
% a.out
文字列：abc は文字列：123 より辞書式順序で後
文字列：abc は文字列：abc と同じ
文字列：ABC は文字列：abc より辞書式順序で前
```

６．５　文字列から数値への変換

文字から数値への変換すると、数値演算が可能になる。

関数 atoi	文字列をint型に変換。
関数 atol	文字列をlong型に変換。

```
 1  /* << c651.c >> */
 2  #include <stdio.h>
 3  #include <strings.h>
 4  main() {
 5    int i;
 6    long j;
 7    double d;
 8    char *word;
 9    /* 文字列からint型への変換。*/
10    word = "12345";
11    printf("文字列：%s  ",word);
12    i = atoi(word);
13    printf("数値：%d ¥n",i);
14    /* 文字列からlong型への変換。*/
15    word = "123456789";
16    printf("文字列：%s  ",word);
17    j = atol(word);
18    printf("数値：%ld ¥n",j);
19  }
```

実行結果

```
% cc c651.c
% a.out
文字列：12345  数値：12345
文字列：123456789  数値：123456789
```

7　課題

7．1　課題１

次の文字列関数を書き、正しく動作することを確認せよ。

関数	strsearch
書き方	strsearch(探索先文字列,探索元文字列);
機能	探索先文字列中に探索元文字列を探し、見つかったらその位置、見つからなかったら-1を返す。

```
1  /* << c711.c >> */
2  /* strsearch関数 */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    int i,j,k;
7    char str1[99],str2[99];
8    int strsearch(char str1[], char str2[]);
9    while( 1 ) {
10     scanf("%s",str1);
11     if( strcmp(str1,"q") == 0 ) { break; }
12     scanf("%s",str2);
13     k = strsearch(str1,str2);
14     printf("str1=%s  str2=%s  k=%d¥n",str1,str2,k);
15   }
16 }
17 int strsearch(char str1[], char str2[]) {
18   int i,j,len1,len2,p;
19   len1 = strlen(str1);
20   len2 = strlen(str2);
21   for( i=0; i<=len1-len2; i++ ) {
22     p = i;
23     for( j=0; j<len2; j++ ) {
24       if(                              ) { p = -1; break; }
25     }
26     if(                                ) { return p; }
27   }
28 }
```

実行結果

```
% cc c711.c
% a.out
abc123defg 123
str1=abc123defg  str2=123  k=3
abc123defg fgh
str1=abc123defg  str2=fgh  k=-1
q
```

７．２　課題２

次の文字列関数を書き、正しく動作することを確認せよ。

関数	strcut
書き方	strcut(抽出先文字列,開始位置,長さ,抽出文字列);
機能	抽出先文字列から文字列を取り出す。

```
1  /* << c721.c >> */
2  /* strcut関数 */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    int len,p;
7    char str1[99],str2[99];
8    int strcut(char str1[], int p, int len, char str2[]);
9    while( 1 ) {
10     scanf("%s%d%d",str1,&p,&len); /* p:開始位置  len:長さ */
11     if( strcmp(str1,"q") == 0 ) { break; }
12     strcut(str1,p,len,str2);
13     printf("str1=%s  p=%d  len=%d  str2=%s¥n",str1,p,len,str2);
14   }
15 }
16 int strcut(char str1[], int p, int len, char str2[]) {
17   int i,j;
18   /* p<lenを満たさない場合、処理不可能。*/
19   if( p >= strlen(str1) ) { str2[0] = '¥0'; return -1; }
20   j = 0;
21   for( i=p; i<p+len; i++ ) {
22     if(                    ) { break; }
23     str2[j] = str1[i];
24     
25   }
26   str2[j] =
27   return 0;
28 }
```

実行結果

```
% cc c721.c
% a.out
abc12345def 3 5
str1=abc12345def  p=3  len=5  str2=12345
abc123 3 3
str1=abc123  p=3  len=3  str2=123
abc123 3 4
str1=abc123  p=3  len=4  str2=123
q 0 0
```

７．３　課題３

次の文字列関数を書き、正しく動作することを確認せよ。

関数	strdiv
書き方	strdiv(分離先文字列,分離文字,分離後文字列);
機能	分離先文字列中の最右端の分離文字で分離し、分離文字以前を分離先文字列と更新し、分離文字以降の文字列を分離後文字列とする。

```
1  /* << c731.c >> */
2  /* strdiv関数 */
3  #include <stdio.h>
4  #include <strings.h>
5  main() {
6    char del,str1[99],str2[99];
7    int strdiv(char str1[], char del, char str2[]);
8    while( 1 ) {
9      scanf("%s",str1);
10     if( strcmp(str1,"q") == 0 ) { break; }
11     printf("str1=%s¥n",str1);
12     del = '.';
13     strdiv(str1,del,str2);
14     printf("str1=%s  del=%c  str2=%s¥n",str1,del,str2);
15   }
16 }
17 int strdiv(char str1[], char del, char str2[]) {
18   int i,j,k,len1;
19   len1 = strlen(str1);
20   for( i=strlen(str1)-1; i>=0; i-- ) {
21     if(                    ) { break; }
22   }
23   if( i < 0 ) {                    return -1; }
24   str1[i] = '¥0'; k = 0;
25   for( j=i+1; j<len1; j++ ) {
26     
27     k++;
28   }
29   str2[k] =            return 0;
30 
31 }
```

実行結果

```
% cc c731.c
% a.out
abc.def
str1=abc.def
str1=abc  del=.  str2=def
abc.def.ghi
str1=abc.def.ghi
str1=abc.def  del=.  str2=ghi
q
```

７．４　課題４

コマンドラインの文字を空白で区切り、文字列として、文字列配列argv[]に読み込むことができる。文字列の個数は変数argcに格納される。簡単なプログラムで確認せよ。

```
1  /* << c741.c >> */
2  #include <stdio.h>
3  main(int argc, char *argv[]) {
4    int i;
5    printf("argc:%d¥n",argc);
6    for( i=0; i<argc; i++ ) {
7      
8    }
9  }
```

実行結果

```
% cc c741.c
% a.out a bb ccc
argc:4
argv[0]:a.out
argv[1]:a
argv[2]:bb
argv[3]:ccc
```