Course Notes Set 6:

COMP1200-001

Introduction to Computing for Engineers and Scientists

C Programming

Chapter 6

Computer Science and Software Engineering

Auburn University

Strings

A string is an array of characters including the terminating null (\0) character.

We have already seen string constants, but we have not used variables to store strings.

"This is a string constant, also called a string literal."
"348 S. College Ave."
"36830-3487"

How is a character or string…
- stored in a variable?
- read into a variable?
- printed from a variable?

Declare Strings

The keyword **`char`** is used to declare both single characters and strings.

Strings are stored in characters arrays. Each character is an element of the array. The last element is the null character, **`\0`**.

The size of character array is the length of the longest string + 1 for **`\0`**.

```
char yes, pound, major[21], color[11];
char address[31], phone[13];
```


Initializing a Single Character

A single character is initialized in an assignment statement using single quotes around the character.

```
char pound;
char yes;
pound = '#';
yes = 'Y';
```

OR

```
char pound = '#';
char yes = 'y';
```

Initializing Strings, or Character Arrays

A character array is initialized with assignment statements character by character with single quotes.

```
char color[11];
color[0] = 'b';
color[1] = 'l';
color[2] = 'u';
color[3] = 'e';
color[4] = '\0';
```

OR

```
char color[11] = "blue";
```

OR

```
char color[11] =
 {'b','l','u','e','\0'};
```

So how is a 30 character address initialized without having to assign one character at a time?

Initializing Strings, or Character Arrays

First, to use the functions that work with strings the file **string.h** must be included.

```
#include <string.h>
```

The library function **strcpy** makes it easy to initialize strings.

```
strcpy(color,"blue");
```

causes the string "blue" to be copied to the memory cells indicated by the array name without brackets, color (which represents an address).

Although **"blue"** does not end in **\0**, C recognized this to be a string and adds **\0** after the **e** when stored in memory.

A string literal is enclosed in double quotes.

strcpy function

```
strcpy(address,"2487 North College Street");
```

Causes the string **"2487 North College Street"** to be copied to the memory cells indicated by the array name address without the brackets.

```
char inColor[11], outColor[11];
strcpy(inColor,"green");
strcpy(outColor,inColor);
```

Assigns the string in **inColor** to **outColor**. **inColor** and **outColor** are the names of arrays without brackets.

Print a single character to a file

```
#include <stdio.h>
int main()
{
  char answer,yes;
  FILE *outfile;
  outfile=fopen("sample.dat","w");
  yes='y';
  answer=yes;
  fputc(answer,outfile);
  fputc(answer,outfile);
  return 0;
}
```

Output: yy

Print a single character to the screen

```
#include <stdio.h>
int main()
{
  char answer;
  char yes;
  yes='y';
  answer=yes;
  putchar(answer);
  return 0;
}                          output: y
```


Print a string to a file

```
#include <stdio.h>
int main()
{
 char inColor[11], outColor[11];
 FILE *outfile;
 outfile=fopen("c:\\a_folder\\sample.dat","w");
 strcpy(inColor,"purple");
 strcpy(outColor,inColor);
 fputs(outColor,outfile);
 fprintf(outfile,"%s\n",outColor);
 return 0;
}                      output:  purplepurple
```


Print a string to the screen

```
#include <stdio.h>
int main()
{
  char inColor[11], outColor[11];
  strcpy(inColor,"purple");
  strcpy(outColor,inColor);
  puts(outColor);
  printf("%s\n",outColor);
  return 0;
}                          output:  purple
                                    purple
```


Print a string to the screen

```
puts(outColor);
```

- **puts** prints element after element of the outColor[] array to the screen it encounters the null character, which it does not print
- at that point it prints a newline character
- it advances the curser to a new line even though it is not explicitly told to
- it is important that the string to be printed ends with a null character

Single character vs one character string

```
char answer, ans[2];
answer='y';
strcpy(ans,"y");
```

answer is NOT equal to **ans**

Memory for char answer	I byte	y	
Memory for char array ans	2 bytes	y	\0

printf Strings

```
#include <stdio.h>
int main()
{
  char address[31];
  strcpy(address,"2487 North College Street");
  printf("[[%s]]\n",address);
  printf("[[%30s]]\n",address);
  printf("[[%-30s]]\n",address);
  printf("[[%30.4s]]\n",address);
  return 0;
}
```

```
[[2487 North College Street]]
output: [[     2487 North College Street]]
        [[2487 North College Street     ]]
        [[                         2487]]
```


printf Strings

```
#include <stdio.h>
int main()
{
  char address[31], city[21], st[3], zip[6];
  strcpy(address,"2487 North College Street");
  strcpy(city,"Auburn");
  strcpy(st,"AL");
  strcpy(zip,"36830");
  printf("%s\n%s, %s  %s\n",address,city,st,zip);
  return 0;
}
```

```
output:  2487 North College Street
         Auburn, AL  36830
```


printf Strings

```
printf("%s\n%s, %s  %s\n",
  "2487 North College Street","Auburn","AL",
  "36830");
```

```
Output:  2487 North College Street
         Auburn, AL  36830
```

Read a string from the keyboard

```
#include <stdio.h>
int main()
{
  char name[16];
  printf("Enter your name: ");
  gets(name);
  puts(name);
  printf("Enter your name: ");
  scanf("%s",name);
  puts(name);
  return 0;
}   Output:  Enter your name: Lee Parker
            Lee Parker
            Enter your name: Lee Parker
            Lee
```


Read a string from the keyboard

```
gets(name);
```

- takes the string input from the keyboard until newline (Enter)
- copies it into the memory location reserved for the array[ ]'s first memory cell
- the newline character is not copied into memory
- the newline character is discarded and a null character is inserted into its place
- the null character terminates the string in memory
- recall that **puts** needs the null character at the end of the string

Read a string from the keyboard

```
scanf("%s",name);
```

- **scanf** with one %s reads just one word NOT an entire line
- %s means to read until a white-space character is encountered

Read a string from the keyboard

```
#include <stdio.h>
int main()
{
  char fname[16],lname[16];
  printf("Enter your name: ");
  scanf("%s%s",fname,lname);
  puts(fname);
  puts(lname);
  return 0;
}
       Output:  Enter your name: Lee Parker
                Lee
                Parker
```

Read a string from a file

```
#include <stdio.h>
#include <string.h>
int main()
{
  char major1[16], major2[16], path[30];
  strcpy(path,"c:\\a_folder\\sample.dat”);
  FILE *infile;
  infile=fopen(path,"r");
  fgets(major1,16,infile);
  puts(major1);
  fgets(major2,16,infile);
  puts(major2);
  return 0;
}
```

```
output:   Mathmetics
          Electrical Engi
```


2-D character arrays

A string is an array of characters.

A way to store a list of strings is in a 2-D character array.

- Each row can be considered a separate string if it is terminated with the null character.

```
char major[4][20];
```

- can collect up to 4 majors
- each major can be up to 19 characters long

2-D character arrays

```
#include <stdio.h>
#include <string.h>  //string.h for strcpy()
int main()
{
  char phrase[4][30];
  strcpy(phrase[0],”This sentence is \n“);
  strcpy(phrase[1],”printed on two lines.“);
  strcpy(phrase[2],”Notice a space is needed “);
  strcpy(phrase[3],”at the end of the phrases.”);
  printf(“%s%s\n%s%s”,
        phrase[0],phrase[1],phrase[2],phrase[3]);
  return 0;
}
Output:
   This sentence is
   printed on two lines.
   Notice a space is needed at the end of the phrases.
```


2-D character arrays

`major[0]` is the address of the first row or the 2-D array.

```
strcpy(major[0],"Electrical Engr");
```

Causes the string **`“Electrical Engr”`** to be copied to the memory cells indicated by the array name address with one bracket, i.e. the address of the first row.

2-D character arrays

```
#include <stdio.h>
int main()
{ char major[4][20];
  int m;
  for(m=0;m<4;m++)
  { printf("Enter a major: ");
    gets(major[m]);
  }
  for(m=0;m<4;m++) puts(major[m]);
  return 0;
}
```

```
Output:  Enter a major: Electrical Engr
         Enter a major: Civil Engineering
         Enter a major: Mathematics
         Enter a major: Chemistry
         Electrical Engr
         Civil Engineering
         Mathematics
         Chemistry
```


Character functions – **isalnum()**

```
#include <stdio.h>
#include <string.h>
int main()
{
  char phrase[50];
  strcpy(phrase,"aNu2837.87");
  puts(phrase);
  if( isalnum(phrase[0]) != 0 )
    printf("NOT all characters: a-z, A-Z, or 0-9 \n");
  else
    printf("All characters: a-z, A-Z, or 0-9 \n");
  return 0;
}
```

```
Output:
   aNu2837.87
   NOT all characters: a-z, A-Z, or 0-9
```



```
#include <stdio.h>
#include <string.h> // string functions
#include <ctype.h>  // functs for checking & changing
                    // case or kind of character
int main()
{ char phrase[50];
  int i, nonChar=0;
  strcpy(phrase,"aNu&2837.87");
  puts(phrase);
  for (i=0;i<strlen(phrase);i++)
  { if( isalnum(phrase[i]) == 0 )
    { nonChar=1;
      putchar(phrase[i]);
    }
    if (nonChar == 1)
     printf(" are NOT characters: a-z,A-Z,or 0-9\n");
    else
      printf("All characters: a-z,A-Z,or 0-9\n");
    return 0;
}
```

Character functions
-**isalnum()**

```
OUTPUT: aNu&2837.87
        &. are NOT characters: a-z,A-Z,or 0-9
```


Character functions – **tolower()**

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main()
{ float num, total=0;
  char done, cont[5];
  done = 'n';
  cont[0] = 'y';
  printf("Sum a list of numbers\n");
  while(cont[0] != done && tolower(cont[0]) != done)
  { printf("\nEnter a number: ");
    scanf("%f",&num);
    total +=num;
    printf("Do you want to enter another number?\n");
    printf("Enter yes or no: ");
    scanf("%s",cont);  }
  printf("Total = %0.2f\n",total);
  return 0;
}
```

Character functions – **tolower()**

```
Sum a list of numbers

Enter a number: 89
Do you want to enter another number?
Enter yes or no: y
Enter a number: 98
Do you want to enter another number?
Enter yes or no: N
Total = 187.00
```


Character functions – atoi()

```
Total some numbers
Enter a number between 1 and 100: 65
Enter a number between 1 and 100: 90
Enter a number between 1 and 100: jh
Entry was not a number between 1 & 100.
Enter a number between 1 and 100: 34
Enter a number between 1 and 100: 0
total = 189
```

If use a char when an integer is needed get:

```
warning: assignment makes integer from
   pointer without a cast
```


Character functions – **atoi()**

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

int main()
{
  int i, total=0, notNum=0;
  char charNum[5];

  printf("Total some numbers");
  printf("\nEnter a number between 1 and 100:");
  gets(charNum);
```



```
while(notNum == 0 && charNum[0] != '0')
{
   for (i=0;i<strlen(charNum);i++)
       if (isdigit(charNum[i]) == 0)  notNum=1;

   if (notNum == 0) total += atoi(charNum);
   else
   {
    printf("Entry was not a number between 1 & 100.\n");
    notNum=0;
   }

   printf("Enter a number between 1 and 100: ");
   gets(charNum);
}
   printf("total = %d\n",total);
   return 0;
```

Character functions – **atoi()**