Lecture 11

Strings

- We have already come across strings e.g. `puts("Hello");`
- Here "Hello" is a literal string or string constant
- We can also have string variables
- In C strings are not a simple data type
- A string is a null-terminated array of `char`
- This means that the end of a string is marked by a "sentinel" character, `'\0'` (ASCII code=0)

h	e	l	l	o	\0	?

Declaring String Variables

- Simplest
  `char message[6];`
- Initialising
  `char message[6]={'H','e','l','l','o',\o'};`
- But this is so common that C provides the shorthand
  `char message[6]="Hello"; or`
  `char message[]="Hello";`
- Which allocates the required 6 bytes automatically

Declaring String Variables

```
My first name is David.
My last name is Hamill.

myname1[0] = 'D' (ASCII 68)
myname1[1] = 'a' (ASCII 97)
myname1[2] = 'v' (ASCII 118)
myname1[3] = 'i' (ASCII 105)
myname1[4] = 'd' (ASCII 100)
myname1[5] = ' ' (ASCII 0)

myname2[0] = 'H' (ASCII 72)
myname2[1] = 'a' (ASCII 97)
myname2[2] = 'm' (ASCII 109)
myname2[3] = 'i' (ASCII 105)
myname2[4] = 'l' (ASCII 108)
myname2[5] = 'l' (ASCII 108)
myname2[6] = ' ' (ASCII 0)
```

- Because the '\0' character marks the end of a string it is ok to have more space than needed
  `char message[80]="Hello";`
- The 74 bytes beyond the '\0' contain rubbish at this point but are not printed

strings1.c

```
/* Example: strings as null-terminated arrays of char */

#include <stdio.h>

main()
{
  char myname1[6] = "David"; /* 5 characters + '\0' */
  char myname2[] = "Hamill"; /* size set automatically */
  int i;

  printf("My first name is %s.\n", myname1);
  printf("My last name is %s.\n\n", myname2);

  for (i = 0; i <= 5; i++)
    printf("myname1[%i] = '%c' (ASCII %i)\n", i, myname1[i],
            myname1[i]);
  putchar('\n');

  for (i = 0; i <= 6; i++)
    printf("myname2[%i] = '%c' (ASCII %i)\n", i, myname2[i],
            myname2[i]);
  putchar('\n');
}
```

Declaring String Variables

- As with all arrays, overrun must be avoided
- This is particularly easy with string manipulation

strings2.c

```
/* Example: inputting strings from keyboard with scanf */

#include <stdio.h>

main()
{
  char word[11];  /* a string, up to 10 characters (+ '\0') */
  char sentence[] = "Very interesting.";

  /* The simplest approach. What might happen if a long word is
  entered? Why? */

  printf("Enter a word, not more than 10 characters: ");
  scanf("%s", word);

  printf("You entered \"%s\". %s\n\n", word, sentence);
}
```

Declaring String Variables

- The `scanf` function allows us to input a string, specifying the maximum number of characters

strings3.c

```
/* Example: inputting strings from keyboard with scanf */

#include <stdio.h>

main()
{
  char word[11];  /* a string, up to 10 characters (+ '\0') */
  char sentence[] = "Very interesting.";

  /* A better approach. What happens if a long word is
  entered? Why? */

  printf("Enter a word, not more than 10 characters: ");
  scanf("%10s", word);
  printf("You entered \"%s\". %s\n\n", word, sentence);
}
```

String Variables as Pointers to `char`

- As with all arrays, we can access individual elements using a pointer. This is declared as
  `char* pc;`
- Note that this only creates a pointer, it does <u>not</u> allocate any memory to it

strings5.c

```
/* Example: string variables as pointers to char */

#include <stdio.h>

main()
{
  char town[] = "Guildford"; /* array of char */
  char *pString;  /* pointer to char */
  int i;

  pString = town + 5;
  puts(town);
  puts(pString);

  for (pString = town, i = 8; i >= 0; i--)
    putchar(pString[i]);
    /* or putchar(*(pString + i)); */
  putchar('\n');

  pString = town; *pString = 'B';
  pString += 5; *pString = '\0';
  puts(town);
}
```

```
Guildford
ford
drofdliuG
Build
```

Manipulating Strings

- Because strings are actually arrays, in general we must operate on individual elements, e.g. we cant do
  `char message[40];`
  `message="Hello";`
- To do this we would probably use a for loop and set each item (`message[i]`) individually
- Because this is a pain and a very common requirement C has a standard library of string manipulation functions `<string.h>` which you can `#include`

`#include <string.h>`

- Some of its useful routines are
  `strcpy(d,s);` - copy string s (source) to string d (destination)
  `strcat(d,s);` - concatenate (join) string s onto the end of string d
  `strcmp(s1,s2);` - compare strings s1 and s2
- String comparison is done on a char-by-char basis
- Starting at position 0 and ending when a '\0' is found in either string
- The chars are compared numerically, using their ASCII codes
- The result is negative if s1<s2, zero if s1==s2 and positive if s1>s2

`#include <string.h>`

`strlen(s);` - gives the length of string s (not counting the '\0')

`strstr(s1,s2);` - gives the position of s2 within s1

strings6.c

```
/* Example: string manipulation using <string.h> library */
/* At the Unix prompt, enter 'man string' for details */

#include <stdio.h>
#include <string.h>

main()
{
  char town[] = "Guildford";
  char county[] = "Surrey";
  char s[80];
  char *pc;
  int n;

  strcpy(s, "Woking");  /* string copy */
  printf("The nearest town to %s is %s.\n\n", town, s);

  strcat(s, " and ");  /* string concatenation */
  strcat(s, town);
  strcat(s, " are both in ");
  strcat(s, county);
  strcat(s, ".\n");
  puts(s);

  if (strcmp(town, county) == 0)  /* string comparison */
    printf("%s is the same as %s.\n", town, county);
  else
    printf("%s is not the same as %s.\n", town, county);

  strcpy(s, "How long is a piece of string?");
  n = strlen(s);  /* string length, excluding the '\0' */
  printf("\n\"%s\" contains %i characters.\n\n", s, n);

  pc = strstr(s, "of");  /* find a string within a string */
  n = pc - s;  /* subtract the pointers */
  printf("\"of\" occurs at position %i in \"%s\"\n", n, s);
}
```

```
The nearest town to Guildford is Woking.

Woking and Guildford are both in Surrey.

Guildford is not the same as Surrey.

"How long is a piece of string?" contains 30 charac

"of" occurs at position 20 in "How long is a piece
```

Printing to a string

- We use `printf` to print to the screen
- Similarly we can use `sprintf` to print to a string
- We can also scan from a string using `sscanf`

sprintf.c

```
/* Example: formatted printing to a string using sprintf */
/* Like printf, but prints to a string instead of standard output */

#include <stdio.h>

main()
{
  char format[40], output[40];
  float x;

  puts("Enter a positive or negative floating point number:");
  scanf("%f", &x);

  /* Assemble format string at run time: */
  sprintf(format, "You entered %s\n", (x >= 0) ? "%6.3f" : "%5.3E");

  /* Generate output string: */
  sprintf(output, format, x);

  puts(output);
}
```

```
Enter a positive or negative floating point number:
1.2
You entered  1.200
```

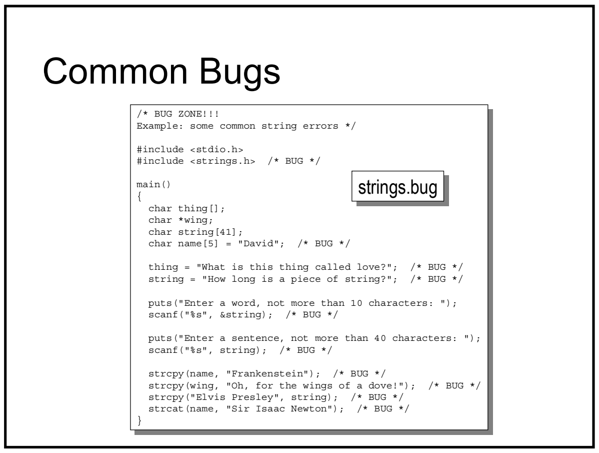
Useful character functions

- The standard library <ctype.h> contains some useful routines to tell us about individual characters
  `islower(c), isupper(c), isdigit(c), isspace(c), ispunct(c)`
- they all return true/false if char c is lower case, upper case……..

analyse2.c

```
/* Example: analysis of text using <ctype.h> library */
/* (Based on analyse.c) */

#include <stdio.h>
#include <ctype.h>

main()
{
  char c;
  int i = 0, lc = 0, uc = 0, dig = 0, ws = 0, pun = 0, oth = 0;

  puts("Type some text (then ENTER):");

  /* Analyse text as it is input: */

  while ((c = getchar()) != '\n')
  {
    if (islower(c))
```

```
      lc++;
    else if (isupper(c))
      uc++;
    else if (isdigit(c))
      dig++;
    else if (isspace(c))
      ws++;
    else if (ispunct(c))
      pun++;
    else
      oth++;
  }

  puts("\nYou typed:");
  printf("\n%i lower case letters\n", lc);
  printf("%i upper case letters\n", uc);
  printf("%i digits\n", dig);
  printf("%i whitespace\n", ws);
  printf("%i punctuation\n", pun);
  printf("%i others\n", oth);
}
```



Common Bugs

strings.bug

```
/* BUG ZONE!!!
Example: some common string errors */

#include <stdio.h>
#include <strings.h>  /* BUG */

main()
{
  char thing[];
  char *wing;
  char string[41];
  char name[5] = "David";  /* BUG */

  thing = "What is this thing called love?";  /* BUG */
  string = "How long is a piece of string?";  /* BUG */

  puts("Enter a word, not more than 10 characters: ");
  scanf("%s", &string);  /* BUG */

  puts("Enter a sentence, not more than 40 characters: ");
  scanf("%s", string);  /* BUG */

  strcpy(name, "Frankenstein");  /* BUG */
  strcpy(wing, "Oh, for the wings of a dove!");  /* BUG */
  strcpy("Elvis Presley", string);  /* BUG */
  strcat(name, "Sir Isaac Newton");  /* BUG */
}
```