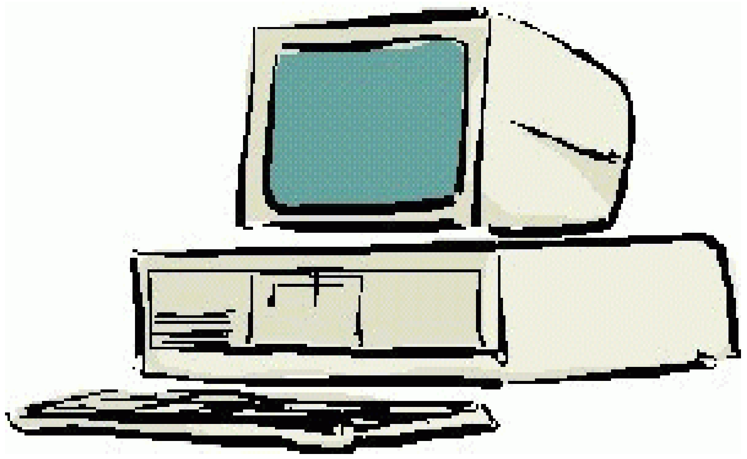
Programming in C



Session 6a

Seema Sirpal
Delhi University Computer Centre

Relationship between Pointers & Arrays

In some cases, a pointer can be used as a convenient way to access or manipulate the data in an array.

Suppose the following declarations are made.

float temperatures[31];
   /* An array of 31 float values, the daily temperatures in a month */

float *temp;   ***/* A pointer to type float */***

Since temp is a float pointer, it can hold the address of a float variable.

Relationship between Pointers & Arrays

The address of the first element of the array temperatures can be assigned to temp in two ways.

```
temp = &temperatures[0];
temp = temperatures;
   /* This is an alternate notation for the first
   element of the array. Same as temperatures = &temperatures[0]. */
```

The temperature of the first day can be assigned in two ways.

```
temperatures[0] = 29.3;
*temp = 15.2;
```

Relationship between Pointers & Arrays

Other elements can be updated via the pointer, as well.

```
temp = &temperatures[0];

*(temp + 1) = 19.0;
   /* Assigns 19.0 to the second element of temperatures */

temp = temp + 9;
   /* temp now has the address of the 10th element of the array */

*temp = 25.0;
   /* temperatures[9] = 25, remember that arrays are zero based,
   so the tenth elementis at index 9. */

temp++;   /* temp now points at the 11th element */

*temp = 40.9;   /* temperatures[10] = 40.9 */
```

Relationship between Pointers & Arrays

Pointers are particularly useful for manipulating strings, which are stored as null terminated character arrays in C

Character Arrays

Strings are stored in C as character arrays terminated by the null character, '\0'.

The array length must be at least one greater than the length of the string to allow storage of the terminator.

String constants or literals are stored internally as a null character terminated character array.

Assigning a character literal to an array is done as follows.

```
char str1[] = "Hello World";
char str2[] = "Goodbye World";
```

Character Arrays

The compiler automatically sizes the arrays correctly. For this example, str1 is of length 12, str2 is of length 14. These lengths include space for the null character that is added at the end of the string.

A character pointer can also be assigned the address of a string constant or of a character array.

```
char *lpointer = "Hello World";
   /* Assigns the address of the literal to lpointer */

char *apointer = str1;
   /* Assigns the starting address of str1 to apointer */

char *apointer = &str1[0];
   /* Assigns the starting address of str1 to apointer */
```

Character Arrays

There is no direct means in the C language to copy one array to another, or one string to another. It must be done either with a standard library function or element wise in a loop. Let's try to copy on string to another.

```
#include <stdio.h>
int main()
{
    char str1[] = "Hello World";
    char str2[] = "Goodbye World";

    str2 = str1;

    return 0;
}
```

Can you see what's wrong with this code. As stated, there is no operation to assign one array to another in C. This code produced this compiler error.

```
error: '=' : cannot convert from 'char [12]' to 'char [14]'
There is no context in which this conversion is possible.
```

Character Arrays

Now let's make another attempt using character pointers.

```
#include <stdio.h>
int main()
{
   char str1[] = "Hello World";
   char str2[] = "Goodbye World";
   char *cpt1;
   char *cpt2;

   cpt1 = &str1[0];
   cpt2 = &str2[0];

   printf("str1 is %s\n",str1);
   printf("str2 is %s\n",str2);
   printf("cpt1 is %s\n",cpt1);
   printf("cpt2 is %s\n",cpt2);

   cpt2 = cpt1;

   printf("str1 is %s\n",str1);
   printf("str2 is %s\n",str2);
   printf("cpt1 is %s\n",cpt1);
   printf("cpt2 is %s\n",cpt2);
   return 0;
}
```

Character Arrays

Results:

```
str1 is Hello World
str2 is Goodbye World
cpt1 is Hello World
cpt2 is Goodbye World
str1 is Hello World
str2 is Goodbye World
cpt1 is Hello World
cpt2 is Hello World
```

As can be seen from the results, all that happened is that the pointer cpt2 was assigned the value of cpt1, that is, the address of str1. The contents of the array str2 were not changed. The only way to copy a string or any array in C is element by element.

Character Arrays

Here is a program that correctly copies one string to another.

```
#include <stdio.h>
int main()
{
    int i;
    char str1[] = "Hello World";
    char str2[] = "Goodbye World";
    printf("str1 is %s\n",str1);
    printf("str2 is %s\n",str2);
    i = 0;
    while ((str2[i] = str1[i]) != '\0') {
        i++;
    }
    printf("str1 is %s\n",str1);
    printf("str2 is %s\n",str2);
    return 0;
}
```

Character Arrays

Results:

```
str1 is Hello World
str2 is Goodbye World
str1 is Hello World
str2 is Hello World
```

Practice Problem

Try re-implementing the above program using pointers in the copy loop.

Hints:

```
    cpt1 = &str1[0];
    cpt2 = &str2[0];
```

Use these pointers in the while loop, remember to dereference.

Solution

```
#include <stdio.h>
int main()
{
    char str1[] = "Hello World";
    char str2[] = "Goodbye World";
    char *cpt1;
    char *cpt2;
    cpt1 = &str1[0];
    cpt2 = &str2[0];
    printf("str1 is %s\n",str1);
    printf("str2 is %s\n",str2);
    while ((*cpt2 = *cpt1) != '\0') {
        cpt2++;
        cpt1++;
    }
    printf("str1 is %s\n",str1);
    printf("str2 is %s\n",str2);
    return 0;
}
```