

PROGRAMMING

CT103
Week 5b

Lecture Content

- Last lecture (Week 5a):
 - Arrays recap
 - Arrays in memory
 - 2D Arrays
 - Example 2D arrays C program
- Today's lecture (Week 5b):
 - What is a string in C
 - How to initialise a string
 - Printing strings
 - Scanning strings
 - Example C program

STRINGS

String Definition

- A string is a collection of characters, i.e. text.
- Specifically, in C strings are defined as an array of characters.
- Examples of strings include:
 - “It’s a nice day!”
 - “My name is Fred”
 - “The temperature outside is 24 degrees”

Strings

- Creating a string:
- If you wanted to create a string in C, you would do something like this:

```
char name[] = "Alex";
```

Why Use Strings?

- Up until now we have only considered numeric, Boolean and single character data.
- Strings are necessary because you will often be manipulating data that consists of text, e.g. names, addresses, etc.

Start with character arrays

- In C there is no variable type “String”.
- This is the case for many other higher level languages.
- We therefore use an array of characters to store a string.
- `char string1[100] = "Hello";`

	string1[0]	string1[1]	string1[2]	string1[3]	string1[4]
string1[100]	'H'	'e'	'l'	'l'	'o'
address:	75F7CC	75F7D0	75F7D4	75F7D8	75F7DC

PRINTING STRINGS

Printing Strings

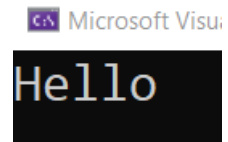
- You could print strings using a for loop as shown below:

```
#include <stdio.h>
void main()
{
    char myString[] = "Hello";
    for (int i = 0; i < 5; i++)
    {
        printf("%c", myString[i]);
    }
    printf("\n\n");
}
```

Printing Strings

- This will work but it is **not recommended**.

```
#include <stdio.h>
void main()
{
    char myString[] = "Hello";
    for (int i = 0; i < 5; i++)
    {
        printf("%c", myString[i]);
    }
    printf("\n\n");
}
```



Strings and Character Arrays

- What is the difference between strings and character arrays?
- A string is terminated with a special character '\0'.
- When you create a string, the character '\0' is automatically put at the end.

Strings in Memory

- `char string1[100] = "Hello";`
- Actually results in memory:

	string1[0]	string1[1]	string1[2]	string1[3]	string1[4]	string1[5]
string1[100]	'H'	'e'	'l'	'l'	'o'	'\0'
address:	75F7CC	75F7D0	75F7D4	75F7D8	75F7DC	75F7E1

So `string[5]` will contain `'\0'` – used to stop processing by any function that processes this string

Printing Strings

- Since all strings end with `'\0'`, you could also print the string using:

```
int i = 0;
while (myString[i] != '\0')
{
    printf("%c", myString[i]);
    i++;
}
```

Printing Strings

- You should simply use %s to print strings.

```
#include <stdio.h>
void main()
{
    char myString[] = "Hello";
    printf("%s",myString);
    printf("\n\n");
}
```

Microsoft Visual Studio

Hello

SCANNING STRINGS

Scanning Strings

- Up until now, you needed to use ‘&’ when scanning in data.
- For example, you would type something like the following for characters (chars):

```
char c;  
scanf_s("%c", &c);
```


Scanning Strings

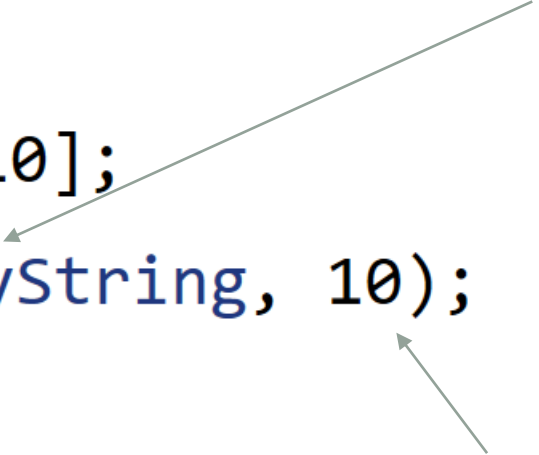
- This is not the case with strings.
- You do not need to use ‘&’ when scanning in strings.
- The reason for this is a bit technical:
 - Char array names decay to pointers in C.
 - The string name already points to the address of the first element in the string.
 - Therefore we don’t need &.

Scanning Strings

- So how do we scan in strings in C?

- Use the following:

```
char myString[10];  
scanf_s("%s", myString, 10);
```



No & symbol!

- Note how you need to specify the character array length!

Scanning Strings

- Lets look at the following example:

```
#include <stdio.h>
void main()
{
    char myString[10]= "hello";
    printf("%s\n", myString);
    printf("Enter a new string: ");
    scanf_s("%s",myString, 10);
    printf("%s\n", myString);
}
```

Scanning Strings

- This outputs the following:

```
#include <stdio.h>
void main()
{
    char myString[10]= "hello";
    printf("%s\n", myString);
    printf("Enter a new string: ");
    scanf_s("%s",myString, 10);
    printf("%s\n", myString);
}
```


Microsoft Visual Studio Debug Console

```
hello
Enter a new string: Hey
Hey
```

Scanning Strings

- `scanf_s` is limited to one single word by default.
- If you enter a space, it will stop scanning.

```
#include <stdio.h>
void main()
{
    char myString[10]= "hello";
    printf("%s\n", myString);
    printf("Enter a new string: ");
    scanf_s("%s",myString, 10);
    printf("%s\n", myString);
}
```

 Microsoft Visual Studio Debug Console


```
hello
Enter a new string: Hi there!
Hi
```

Scanning Two Words

- You can do the following if you want to scan two words.

```
char firstName[10];  
char surname[10];
```

```
printf("Enter first name: ");  
scanf_s("%s", firstName, 10);  
printf("Enter last name: ");  
scanf_s("%s", surname, 10);
```

 C:\Users\Karl\source\repos\CT103_C_Programming

```
Enter first name: Bob  
Enter last name: Smith
```

Strings with Two Words

- This does not mean that you cannot have a string with spaces in it.

```
char myName[10] = "Bob Smith";  
printf("My name is %s.\n", myName);
```

 C:\Users\Karl\source\repos\CT103_C_Programming\Debug\CT103_C_Program

```
My name is Bob Smith.
```

Scanning Two Words

- If you do want to scan two words into one string, you can do the following:


```
scanf_s("%[^\\n]*c", myString, 10);
```

- The `[^\\n]` tells scanf to keep reading characters until a new line is entered (`\\n`).
- The `%*c` remove the new line from the input buffer.

Scanning Two Words

- Lets see this work in a C program:

```
char myString[10]= "hello";  
printf("%s\n", myString);  
printf("Enter a new string: ");  
scanf_s("%[^\\n]*c",myString, 10);  
printf("%s\n", myString);
```

 Microsoft Visual Studio Debug Console

```
hello  
Enter a new string: Hi there!  
Hi there!
```

EXAMPLE PROBLEMS

Employee Name Scanner

- You are writing software to read in employee names. Write a program that:
 - Reads in employee names as strings.
 - The program should stop reading names if the character '!' is entered.
 - Count how many employees have names beginning with 'b' or 'B'.
 - Print the answer to the screen.

Employee Name Scanner

- Go to C program solution.


Employee Name Scanner

- The following code will work:

```
#include <stdio.h>
void main()
{
    int count = 0;
    char newName[10] = "Alex";
    while (newName[0]!='!') {
        printf("Enter a name: ");
        scanf_s("%[^\\n]*c", newName, 10);
        if (newName[0]=='b' || newName[0] == 'B') {
            count++;
        }
    }
    printf("%s is not a name.\\n", newName);
    printf("There are %d names beginning with b/B.", count)
}
```

Employee Name Scanner

- C Program Output:

 Microsoft Visual Studio Debug Console

```
Enter a name: Alex
Enter a name: Brian
Enter a name: Breda
Enter a name: brenda
Enter a name: beth
Enter a name: Bert
Enter a name: Conor
Enter a name: Clair
Enter a name: !
! is not a name.
There are 5 names beginning with b/B.
```