

# PROGRAMMING

---

CT103  
Week 6a

# Lecture Content

- Last lecture (Week 6a):
  - What is a string in C
  - How to initialise a string
  - Printing strings
  - Scanning strings
  - Example C program
- Today's lecture (Week 6b):
  - Length of a string
  - Insert data in a string
  - String functions
  - Example C program

# STRING LENGTH

---

# String Recap

- A string is a collection of characters, i.e. text.
- Specifically, in C strings are defined as an array of characters.
- You can create a string in C as follows:

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

# String Length

- The **length** of a string is always the number of characters up to, but not including, the string terminator.
- By string length we really mean the number of characters actually used.
- The length of the following string is **9**:
  - “August 10\0”

# Size of String

- Any string should be big enough to hold the text you need to **put into it** PLUS 1 more for the null character

```
// can hold up to 3 characters + null character  
// size determined by [4]  
char str1[4] = "One";
```

```
// can hold up to 3 characters + null character  
// size determined when it is initialised with "Two"  
char str2[] = "Two";
```

```
// can hold up to 99 characters + null character  
// even though we only use 6 at initialisation  
char str3[100] = "Three";
```

# Get Length of String

- We could count the length of the string ourselves:

```
char string1[100] = "This is some random text";

int len = 0;

while (string1[len] != '\0')
{
    len++;
}

printf("Length of string = %d \n", len);
```

- Try out this code yourself!

# Get Length of String

- It is much faster if we use the `strlen()` function to get the length of the string.
- `strlen()` does what the previous example does.
- In order to use `strlen()`, we need to include the “string.h” library at the beginning of the program.
- This is a library of string functions.



# Get Length of String using strlen()

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

**Don't forget this!**



```
void main()
{
    char string1[100] = "This is some random text";

    int len = strlen(string1);

    printf("Length of string = %d \n", len);
}
```

# Get Length of String using strlen()

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

**Don't forget this!**

```
void main()
{
    char string1[100] = "This is some random text";

    int len = strlen(string1);

    printf("Length of string = %d \n", len);
}
```

Microsoft Visual Studio Debug Console

```
Length of string = 24
```

# Length of a String Summary

- A string is just an array of characters.
- To use a string it must be **terminated** properly – this means the last character in the array must be the null character ‘\0’.
- Functions that return the length of a string don’t count the null character (even though they return it), so you always have to allocate an array of size 1 more than the number of characters you want to store.
- The length of the following string is 9:
  - “August 10”
- However, you would need to allocate an array of characters of size 10 to hold it!
- Usually you just allocate **plenty !**

# DATA INTO STRINGS

---

# Putting Data into Strings

- In a previous lecture, we talked about setting strings using `scanf_s`, e.g.

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

- How would we set a string without scanning in text?
- Can I simply write the following?

```
myString = "hi";
```

# Putting Data into Strings

- Can I simply write the following? `myString = "hi";`
- No, this won't work. `myString = "hi";`
- You need to use `strcpy_s()` from the `string.h` library that we mentioned before.

# Strcpy\_s()

- See the following example that uses strcpy\_s()

```
#include <stdio.h>
#include <string.h>
void main()
{
    char newName[50] = "Bobbb Smith";
    printf("My name was %s.\n", newName);
    strcpy_s(newName, 50, "Bob Smith");
    printf("My name is %s.\n", newName);
}
```

# Strcpy\_s()

- Produces the following output:

```
#include <stdio.h>
#include <string.h>
void main()
{
    char newName[50] = "Bobbb Smith";
    printf("My name was %s.\n", newName);
    strcpy_s(newName,50,"Bob Smith");
    printf("My name is %s.\n", newName);
}
```

Microsoft Visual Studio Debug Console

```
My name was Bobbb Smith.
My name is Bob Smith.
```



# STRING FUNCTIONS

---

# Common String functions

- `Strcpy_s()` Copy one string to another (**seen already**)
  - `Strncpy_s()` Copy n characters from one string to another
- `Strcat_s()` Link together (concatenate) two strings
  - `Strncat_s()` concatenate n characters from two strings
- `strcmp()` Compare two strings
  - `strncmp()` Compare n characters from two strings

# Strncpy\_s()

- Strncpy\_s()
- Copy n characters from one string to another.

```
char tName[] = "Tommy";  
char newName[50] = "Bobb Smith";  
printf("My name is %s.\n", newName);  
strcpy_s(newName, 50, "Bob Smith");  
printf("My name is %s.\n", newName);  
strncpy_s(newName, 50, tName, 3);  
printf("My name is %s.\n", newName);
```

```
My name is Bobbb Smith.  
My name is Bob Smith.  
My name is Tom.
```

# Strcat\_s()



concatenate

/kənˈkætɪneɪt/

verb FORMAL • TECHNICAL

link (things) together in a chain or series.

"some words may be concatenated, such that certain sounds are omitted"

- Strcat\_s()
- Strcat\_s() Link together (concatenate) two strings

```
char myName[50] = "Tommy";  
printf("\n\nMy name is %s.\n", myName);  
strcat_s(myName, 50, " Smith");  
printf("My name is %s.\n", myName);
```

```
My name is Tommy.  
My name is Tommy Smith.
```

# Strncat\_s()

- Strncat\_s()
- Strncat\_s() concatenate n characters from two strings

```
char myName[50] = "Tommy";  
printf("\n\nMy name is %s.\n", myName);  
strcat_s(myName,50," Smith");  
printf("My name is %s.\n", myName);  
strncat_s(myName, 50, " Smithyyyy",7);  
printf("My name is %s.\n", myName);
```

```
My name is Tommy.  
My name is Tommy Smith.  
My name is Tommy Smith Smithy.
```

# Strcmp()

- Compare two strings
- Strcmp() will return 0 if both strings are the same.

```
char fName1[] = "Tom";  
char fName2[] = "Tim";  
if (strcmp(fName1, "Tom") == 0) {  
    printf("\n\nYou found Tom.\n");  
}  
else {  
    printf("\n\nKeep looking.\n");  
}
```

C# Microsoft Visual Studio Debug Console

You found Tom.

# Strcmp()

- Compare two strings
- Strcmp() will return 0 if both strings are the same.

```
char fName1[] = "Tom";  
char fName2[] = "Tim";  
if (strcmp(fName2, "Tom") == 0) {  
    printf("\n\nYou found Tom.\n");  
}  
else {  
    printf("\n\nKeep looking.\n");  
}
```

Microsoft Visual Studio Debug Console

Keep looking.

# Strncmp()

- Compare n characters from two strings
- Strncmp() will return 0 if first n chars of both strings are the same.

```
char fName1[] = "Tom";  
char fName2[] = "Tim";  
if (strncmp(fName1, fName2, 1) == 0) {  
    printf("\n\nSame first letter.\n");  
}  
else {  
    printf("\n\nDifferent first letter.\n");  
}
```

Microsoft Visual Studio Debug Console

Same first letter.



# Strncmp()

- Compare n characters from two strings
- Strncmp() will return 0 if first n chars of both strings are the same.

```
char fName1[] = "Tom";  
char fName2[] = "Tim";  
if (strncmp(fName1, fName2, 2) == 0) {  
    printf("\n\nSame first 2 letter.\n");  
}  
else {  
    printf("\n\nDifferent first 2 letter.\n");  
}
```

Microsoft Visual Studio Debug Console

Different first 2 letter.

# Note on last weeks example

- We used **newName[0]!='!'**
- We could also use **strncmp()**

```
#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)
}
```

# Note on last weeks example

- See strcmp()

```
#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)
}
```

```
#include <stdio.h>
#include <string.h>
void main()
{
    int count = 0;
    char newName[10] = "Alex";
    while (!strcmp(newName, "!")) {
        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);
}
```

# EXAMPLE PROBLEMS

---

# Employee Name Comparison

- You are writing more software to read in employee names. Write a program that:
  - Reads in 3 employee names as strings.
  - Check the first letter against the target name “Bobby”.
  - If the name also begins with the letter ‘B’, check and see if the full names are the same.

# Employee Name Comparison

- Go to C program solution.

# Employee Name Comparison

- The following code will work:

```
#include <stdio.h>
#include <string.h>
void main()
{
    int count = 0;
    char targetName[10] = "Bobby";
    char newName[10] = "Alex";

    for (int i = 0; i < 3; i++) {
        printf("Enter a name: ");
        scanf_s("%[^\\n]*c", newName, 10);

        if (strncmp(newName, targetName, 1) == 0) {
            printf("Same first letter, checking full name\\n");

            if (strcmp(newName, targetName) == 0) {
                printf("We have a match!\\n");
            }
            else {
                printf("Not a match!\\n");
            }
        }
        else {
            printf("Definetly not a match\\n");
        }
    }
}
```

# Employee Name Comparison

- C Program Output:

```
Microsoft Visual Studio Debug Console  
Enter a name: Clair  
Definetly not a match  
Enter a name: Brenda  
Same first letter, checking full name  
Not a match!  
Enter a name: Bobby  
Same first letter, checking full name  
We have a match!
```