Strings in C

Strings

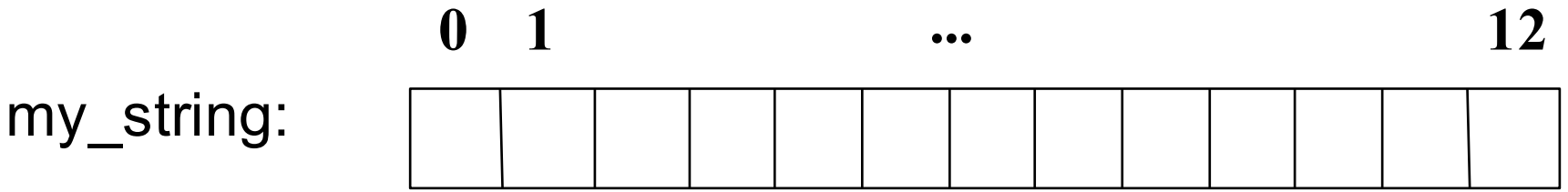
- There is no built in string type in the C language.

- Thus, we use an array of char instead:

```
char my_string [13];
```

Strings, cont.

- my_string looks like this:

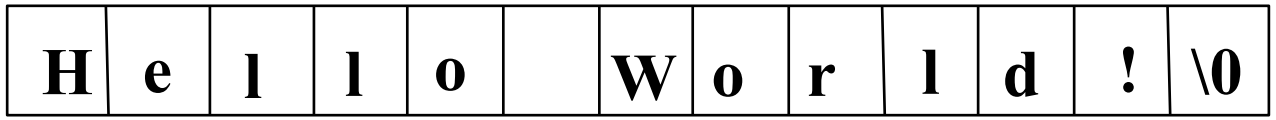


- And has enough space to store 12 characters and a string terminator
- Like all arrays in C, this character array starts with index 0 (zero)

- We can initiate a string in the following manner:

```
char my_string [] = "Hello World!"
```

- my_string:



- The string terminating character null that is written as '\0'

- This type of assignment statement can only be used in a variable declaration.

- We cannot do the following:

```
char my_string [13];
my_string = "Hello World"; //No No No...
```

- Instead, we can use a library function from <string.h> called strcpy:

```
char my_string [13];

strcpy(my_string,"Hello World");
```

- strcpy copies from a source string into a destination string and adds \0 to the end (if there is room!!)
- This is actually a better way to use a string than the previous examples - we're not ready to say why yet :(
- *strcpy requires that we include string.h in our .c file*

- Or, if we're masochists we can assign one character at a time:

```
char my_string [13];
my_string[0] = 'H'; my_string[1] = 'e';
my_string[2] = 'l'; my_string[3] = 'l';
my_string[4] = 'o'; my_string[5] = ' ';
my_string[6] = 'W'; my_string[7] = 'o';
my_string[8] = 'r'; my_string[9] = 'l';
my_string[10] = 'd'; my_string[11] = '!';
my_string[12] = '\0';
```

- Yet another way:

```
char my_string [13] = {'H','e','l','l','o',
      ' ','W','o','r','l','d','!','\0'};
```

- Commonly used library functions from string.h:

- **strcpy** - copy a string
- **strlen** - get string length
- **strcat** - concatenate two strings
- **strcmp** - compare two strings
- **strchr** - string scanning operation
- **strncat** - concatenate one string with part of another
- **strncmp** - compare parts of two strings
- **strncpy** - copy part of a string
- **strrchr** - string scanning operation

strcmp

- Using strcmp:

```
char string_1 [50] = "Mamma Mia";
char string_2 [50] = "Mamma Cass";

if (strcmp(string_1,string_2) > 0) {
   printf("Mia\n");
} else if (strcmp(string_1,string_2) < 0) {
   printf("Cass\n");
} else {
   printf("Same same\n");
}
```

strncmp

- Using strncmp:

```
char string_1 [50] = "Mamma Mia";
char string_2 [50] = "Mamma Cass";


if (strncmp(string_1,string_2,5) > 0) {
    printf("Mia\n");
} else if (strncmp(string_1,string_2,5) < 0) {
    printf("Cass\n");
} else {
    printf("Same same\n");
}
```

strlen

- Using strlen:

```
char string_1 [50] = "Mamma Mia";
char string_2 [50] = "Mamma Cass";

if (strlen(string_1) > strlen(string_2)) {
    printf("Mia\n");
} else if (strlen(string_1) < strlen(string_2)) {
    printf("Cass\n");
} else {
    printf("Same length\n");
}
```

strlen, cont.

- strlen is often used to control an iteration when traversing a string:

```
char string_1 [50] = "Mamma Mia";

for (int i = 0;i<strlen(string_1);i++)  {
    printf("%c ",string_1[i]);
}
printf("\n");
```

strcat

- Using strcat:

```
char string_1 [50] = "Mamma Mia";
char string_2 [50] = "Mamma Cass";


printf("%s\n",strcat(string_1,string_2));
printf(“%s\n”,string_1);
```

Bookchapter on strings

- An introduction to the string.h library together with code examples can be found at:

https://en.wikibooks.org/wiki/A_Little_C_Primer/C_String_Function_Library

Inputing strings from the keyboard (stdin)

- There are a number of functions that can be used, for example:
- scanf ( ) Read formatted data from stdin.
- getchar( ) Read a single character from stdin.
- gets( ) Read a line from stdin.

- gets ( ) is dangerous and should absolutely not be used.

Use scanf

- For the time being use scanf:

```
char string_1 [256];

printf("Enter a string: ");
scanf("%255s",string_1);
printf("Your string is: %s\n",string_1);
```

- The format %255s copies max 255 characters from stdin and adds the '\0' terminator