Introduction to group of words and filter on condition basis

Organised & Supported by **RuggedBOARD**

Agenda

- Defining a String
- Printing String
- ASCII Code
- Length of a String
- Changing case UtoL of a String
- Reverse a String
- String lib Functions
- Typedef
- Typedef Vs #define
- Enumeration

Defining a String

A string is a sequence of text characters that can become a value for a string variable.
Array of characters which is terminated by a null character '\0'.

How to initialize strings?

```
char c[] = "abcd";

char c[50] = "abcd";

char c[] = {'a', 'b', 'c', 'd', '\0'};

char c[5] = {'a', 'b', 'c', 'd', '\0'};
```

```
include<stdio.h>

int main()
{
	//const char name[10]="phytec";
	//char name[10]="phytec";
	char name[10];
	name = "phytec";

	name[2] = 'z';

	printf("%s\n",name);
	return 0;
}
```

Assigning Values to Strings :

```
1.char c[] = "abcd";
2.char c[100];
c = "C programming";  // Error! array type is not assignable.
```

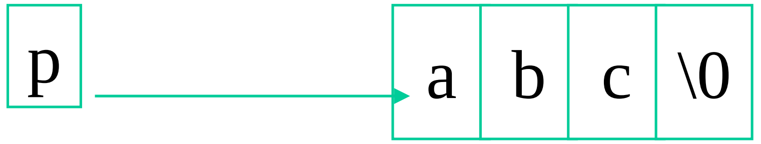
Printing String

Reading strings: %s format

%s reads a string into a character array given the array name or start address.
It ends the string with '\0'

```
#include<stdio.h>
Int  main()
{
   char name[25];
   scanf("%s", name);
   printf("Name = %s \n", name);
   return 0;
}
```

A string constant is treated as a pointer
Its value is the base address of the string
char *p = "abc";



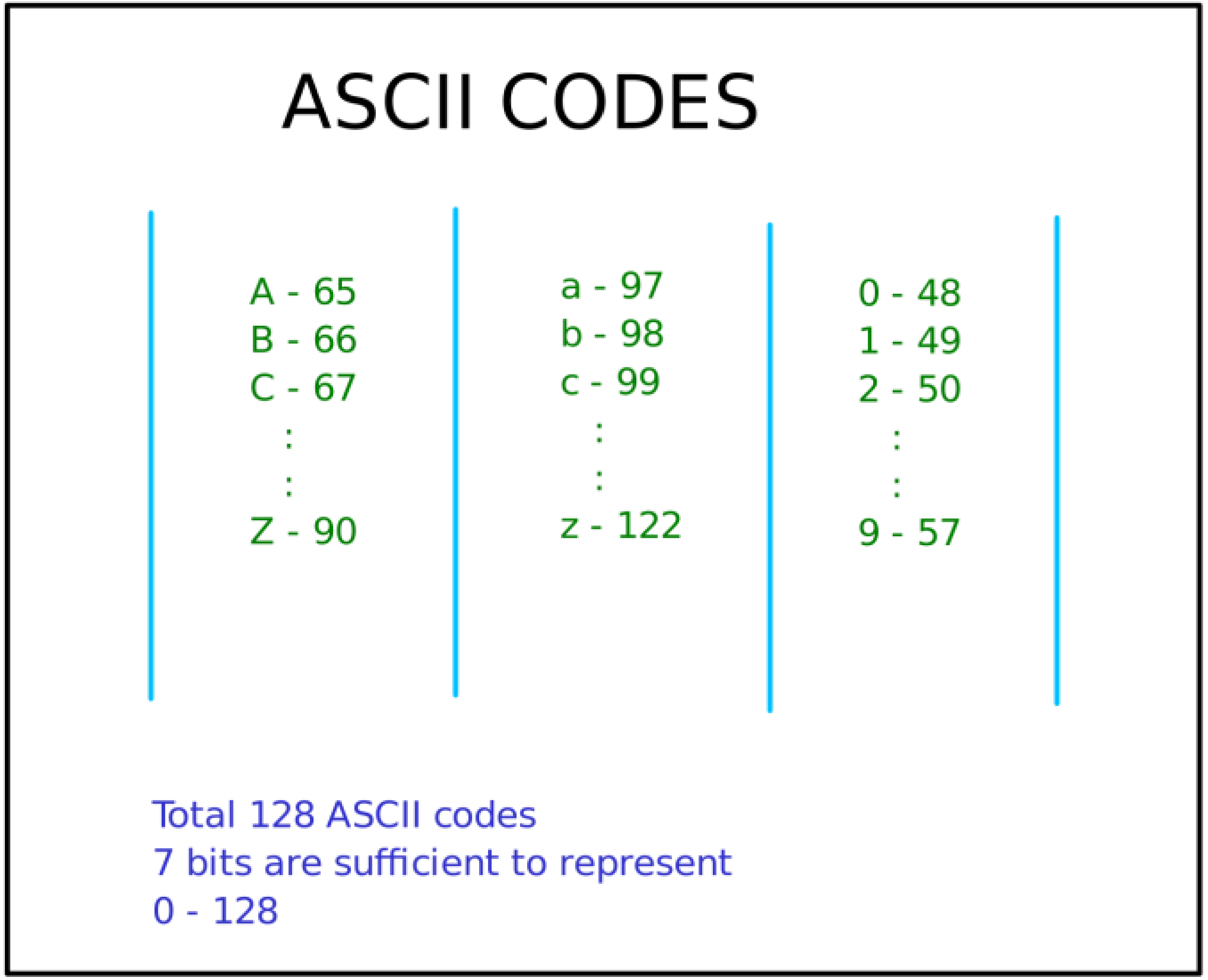
printf ("%s %s\n",p,p+1); /* abc bc is printed */

Printing String

```
#include<stdio.h>

int main()
{
	char name[20] = "Phytec Embedded";

	for(int i=0;name[i] != '\0';i++)
	{
		printf("%c",name[i]);
	}
	printf("\n");
	printf("%s\n",name);
	return 0;
}
```

```
#include<stdio.h>

int main()
{
	char a;

	while((a= getchar() != 10))
		printf("%c",a);
	return 0;
}
```

ASCII Codes



ASCII CODES

A - 65	a - 97	0 - 48
B - 66	b - 98	1 - 49
C - 67	c - 99	2 - 50
:	:	:
:	:	:
Z - 90	z - 122	9 - 57

Total 128 ASCII codes
7 bits are sufficient to represent
0 - 128

Length of a string

```
#include<stdio.h>

int string_len(char *);

int main()
{
	int len;
	len = string_len("Welcome");
	printf("The length of the string is : %d\n",len);
	return 0;
}


int string_len(char string[])
{
	int len;

	for(len=0 ; string[len] != '\0';len++)
	{}

	return len;
}
```

Changing case UtoL of a string

```
#include<stdio.h>

void string_case_ul(char *);

int main()
{
	string_case_ul("WELCOME");
	return 0;
}

void string_case_ul(char* str)
{
	int i;
	char a[10];
	for(i=0 ; *str != '\0';str++,i++)
	{
		a[i] = *str + 32;
	}

	printf("The case changed string is '%s'\n",a);
}
```

Reverse a string

```
#include <stdio.h>
#include <string.h>
void reverseString(char *);
int main()
{
  char str[512];
  scanf("%s", str);
  reverseString(str);
  printf("\nString After Reverse: %s\n", str);
  return 0;
}
void reverseString(char str[])
{
  int n = strlen(str);

  for (int i = 0; i < n / 2; i++)
  {
    char ch = str[i];
    str[i] = str[n - i - 1];
    str[n - i - 1] = ch;
  }
}
```

String lib functions

String functions defined in the header file “string.h”

strcpy()	copies a string to another string
strcat()	concatenates two strings
strrev()	reversed strings of a string
strcmp()	compare two strings
strlwr()	converts into lower case
strupr()	converts into upper case
strstr()	substring of the given string

String lib functions

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

int main()
{
        char name[30] = "Phytec Embedded";
        char name1[30];
        char * ptr;
        ptr = strcpy(name1,name);
        printf("name is %s\n",name);
        printf("name1 is %s\n",name1);

        return 0;
}
```

```
#include <stdio.h>
#include <string.h>
int main()
{
    char destination[] = "Hello ";
    char source[] = "World!";
    strcat(destination,source);
    printf("Concatenated String: %s\n", destination);
    return 0;
}
```

Typedef

The C programming language provides a keyword called **typedef**, which you can use to give a type a new name.

typedef unsigned char BYTE;
After this type definition, the identifier BYTE can be used as an abbreviation for the type **unsigned char, for example..**
BYTE b1, b2;

```
#include <stdio.h>
#include <string.h>
typedef struct Books
{
  char title[50];
  char author[50];
  char subject[100];
  int book_id;
} Book;
int main( )
{
   Book book;
   strcpy( book.title, "C Programming");
   strcpy( book.author, "Nuha Ali");
   strcpy( book.subject, "C Programming Tutorial");
   book.book_id = 6495407;
   printf( "Book title : %s\n", book.title);
   printf( "Book author : %s\n", book.author);
   printf( "Book subject : %s\n", book.subject);
   printf( "Book book_id : %d\n", book.book_id);
   return 0;
 }
```

Typedef vs #define

#define is a C-directive which is also used to define the aliases for various data types similar to **typedef** but with the following differences −

- **typedef** is limited to giving symbolic names to types only
  where as **#define** can be used to define alias for values as well.
  you can define 1 as ONE etc.
- **typedef** interpretation is performed by the compiler
  whereas **#define** statements are processed by the pre-processor.

```
#include <stdio.h>
#define TRUE 1
#define FALSE 0
int main( )
{
  printf( "Value of TRUE : %d\n", TRUE);
  printf( "Value of FALSE : %d\n", FALSE);
  return 0;
}
```

Enumeration

Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

```
#include<stdio.h>

enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};

int main()
{
        enum week day;
        day = Wed;
        printf("%d\n",day);
        return 0;
}
```

Thank You