

Strings

CHAPTER 32



SURESH TECHS

C PROGRAMMING COURSE

String

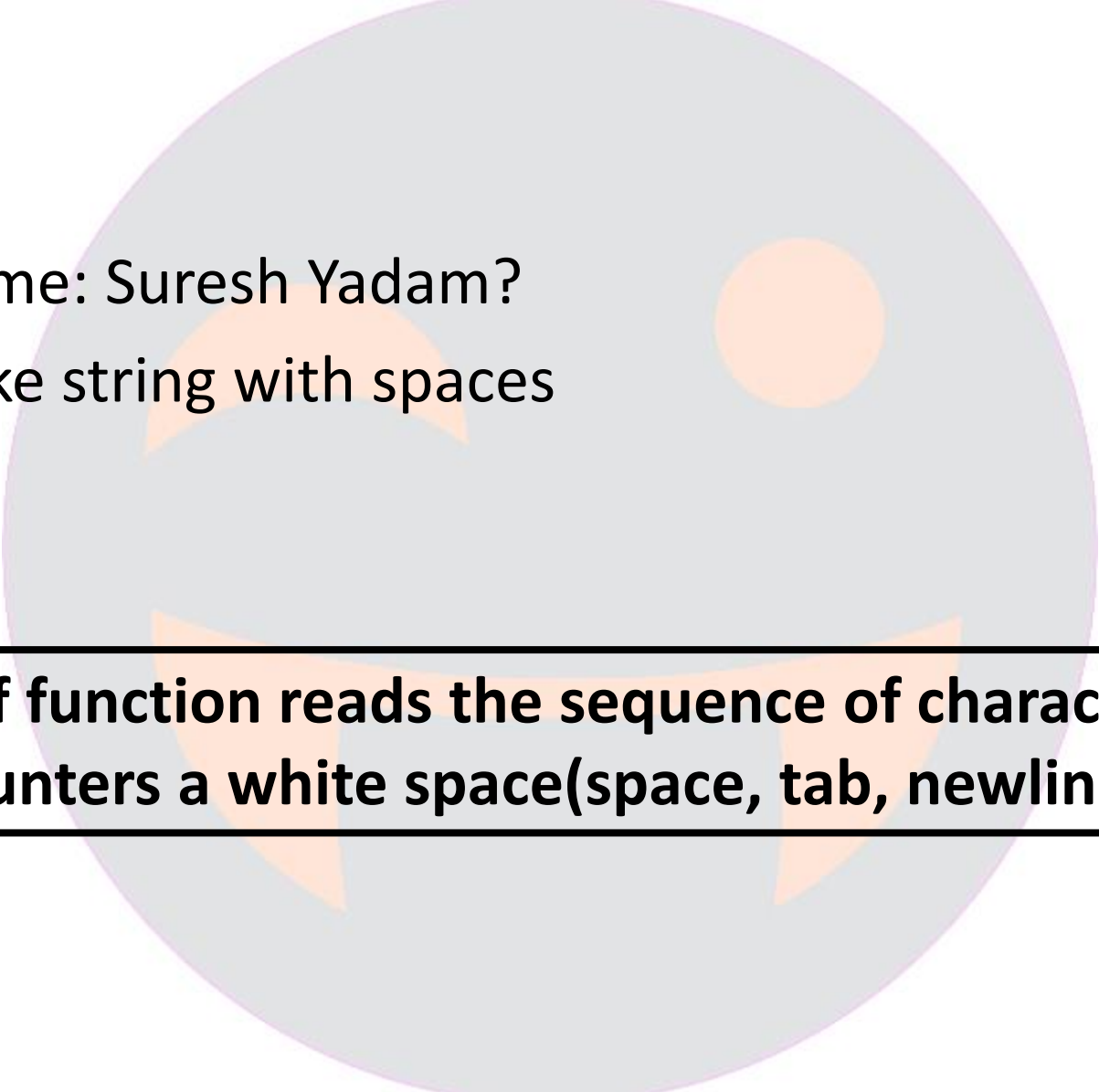
There is an issue using character arrays for strings

- Collection of **characters** is called string(SURBALI)

```
#include<stdio.h>
int main() {
    int noOfBores = 10;
    float passPercentage = 30.26;
    char enemy[] = "SURBALI";
    printf("%d\n", noOfBores);
    printf("%f\n", passPercentage);
    printf("%s", enemy);
    return 0;
}
```



SURBALI

- 
- Let us take name: Suresh Yadam?
 - Not able to take string with spaces

Note: scanf function reads the sequence of characters until it encounters a white space(space, tab, newline etc)

Reading strings from terminal



```
#include<stdio.h>
int main(){
    char name[10];
    printf("Enter name: ");
    /*scanf terminates it's input on the first white
    space it finds. A white space includes blanks, tabs,
    carriage returns, form feeds and new line*/
    scanf("%s",name);
    printf("Welcome %s",name);
    return 0;
}
```

Taking strings with spaces – gets(char array)

fgets()

```
#include<stdio.h>
int main() {
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Welcome %s", name);
    return 0;
}
```

```
#include<stdio.h>
int main() {
    char name[20];
    printf("Enter name: ");
    gets(name);
    printf("Welcome %s", name);
    return 0;
}
```

gets() function is dangerous as it keeps on reading data until it encounters a new line character without even checking the maximum capacity of the character array – buffer overflow problem

```
#include<stdio.h>
int main() {
    char name[20];
    printf("Enter name: ");
    gets(name);
    printf("Welcome %s", name);
    return 0;
}
```

```
#include<stdio.h>
int main() {
    char name[20];
    printf("Enter name: ");
    fgets(name, 10, stdin);
    printf("Welcome %s", name);
    return 0;
}
```

```
char* fgets(char* __restrict __Buf, int _MaxCount,  
FILE* __restrict __File)
```

- char *fgets(char *str, int n, FILE *stream)
- str : Pointer to an array of chars where the string read is copied.
- n : Maximum number of characters to be copied into str (including the terminating null-character).
- *stream : Pointer to a FILE object that identifies an input stream(stdin)
- returns : the function returns str

```
#include<stdio.h>  
int main() {  
    char name[20];  
    printf("Enter name: ");  
    fgets(name,10,stdin);  
    printf("Welcome %s",name);  
    return 0;  
}
```

Write a program to take feedback from suresh techs youtube channel – maximum 100 characters

```
#include<stdio.h>
int main() {
    char feedback[100];
    printf("Enter your feedback about suresh techs
youtube channel: ");
    fgets(feedback,100,stdin);
    printf("Feedback: %s",feedback);
    return 0;
}
```

It keep on reading **until** new line character encountered or **maximum limit** of character array.

puts to print/display the string to the console

- int puts(char *string)

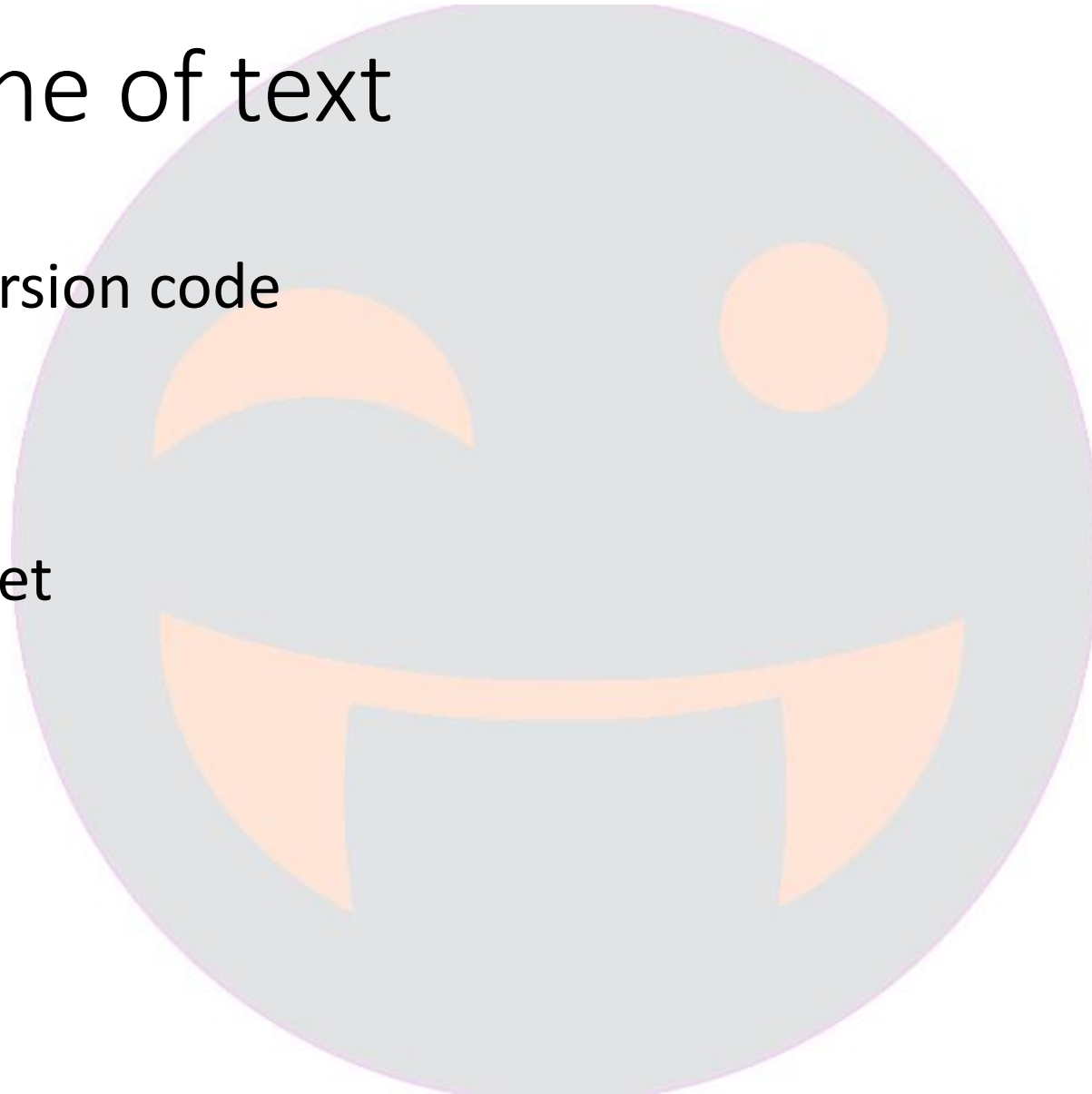
```
#include<stdio.h>
int main(){
    char feedback[100];
    printf("Enter your feedback about suresh techs
youtube channel: ");
    fgets(feedback,100,stdin);
    printf("Feedback: ");
    puts(feedback);
    return 0;
}
```

Write a program to display 5 of your friends

```
#include<stdio.h>
int main() {
    char friends[5][30];
    for(int i =0;i<5;i++){
        printf("Enter %d name ",i+1);
        fgets(friends[i],30,stdin);
    }
    for(int i=0;i<5;i++){
        puts(friends[i]);
    }
    return 0;
}
```

Reading line of text

- edit set conversion code
- `%[. .]`
- `%[^\\n]s`
- `[]` called scanset
- `%[^h]`



Built in string functions - <string.h>

Function	Description
strlen(string_name)	Returns the length of string name
strcpy(destination, source)	Copies the contents of source string to destination string
strcat(first_string, second_string)	Concatenates or joins first string with second string. The result of the string is stored in first string.
strcmp(first_string, second_string)	Compares the first string with second string. If both strings are same, it returns 0.
strrev(string)	Returns reverse string.
strlwr(string)	Returns string characters in lowercase.
strupr(string)	Returns string characters in uppercase.

strlen(string_name)

Suresh, neeku nickname
pettali ra

```
#include<stdio.h>
int main() {
    char friend1[30];
    char friend2[30];
    printf("Enter your first friend: ");
    fgets(friend1,30,stdin);
    printf("Enter your second friend: ");
    fgets(friend2,30,stdin);
    if(strlen(friend1)-1>15){
        friend1[strlen(friend1) - 1] = '\\0';
        printf("%s, neeku nickname pettali ra",friend1);
    }
    if(strlen(friend2)-1>15){
        friend2[strlen(friend2) - 1] = '\\0';
        printf("%s, neeku nickname pettali ra",friend2);
    }
    return 0;
}
```

strcpy(destination, source)

```
#include<stdio.h>
int main() {
    char friend1[30]="Suresh Yadam";
    char friend2[30]="John";
    printf("Friend1 = %s, Friend2 = %s\n",friend1,friend2);
    strcpy(friend1,friend2);
    printf("Friend1 = %s, Friend2 = %s",friend1,friend2);
    return 0;
}
```

```
Friend1 = Suresh Yadam, Friend2 = John
Friend1 = John, Friend2 = John
```

strcat(first_string, second_string)

- The strcat(first_string, second_string) function concatenates two strings and result is returned to first_string.

```
#include<stdio.h>
int main(){
    char friend1[30]="Suresh Yadam";
    char friend2[30]="John";
    printf("Friend1 = %s, Friend2 = %s\n",friend1,friend2);
    strcat(friend1,friend2);
    printf("Friend1 = %s, Friend2 = %s",friend1,friend2);
    return 0;
}
```

```
Friend1 = Suresh Yadam, Friend2 = John
Friend1 = Suresh YadamJohn, Friend2 = John
```

strcmp(first_string, second_string)

- The strcmp(first_string, second_string) function compares two strings and **returns 0 if both strings are equal.**
- We can also use pointers to compare strings which will be discussed later 😊

Couple Game

1. Where did you both met for the first time?

Boy: Coffee shop

Girl: Shopping mall

If both of them gives same answer then they get one point

```
#include<stdio.h>
int main(){
    char boyAnswer[30];
    char girlAnswer[30];
    printf("Where did you both met for the first time?\n");
    printf("Boy: ");
    fgets(boyAnswer,30,stdin);
    printf("Girl: ");
    fgets(girlAnswer,30,stdin);
    if(strcmp(boyAnswer,girlAnswer)==0){
        printf("Both are correct");
    }else{
        printf("Ohh no, you are wrong!");
    }
    return 0;
}
```



strrev(string)

- The strrev(string) function returns **reverse of the given string**.

```
#include<stdio.h>
int main() {
    char name[30]="Suresh Techs";
    printf("Before: %s\n",name);
    strrev(name);
    printf("After: %s",name);
    return 0;
}
```

```
Before: Suresh Techs
After: shceT hseruS
```

strlwr(string)

- The strlwr(string) function returns string characters in lowercase.

```
#include<stdio.h>
int main() {
    char name[30]="Suresh Techs";
    printf("Before: %s\n",name);
    strlwr(name);
    printf("After: %s",name);
    return 0;
}
```

```
Before: Suresh Techs
After: suresh techs
```

strupr(string)

- The strupr(string) function returns string characters in uppercase.

```
#include<stdio.h>
int main() {
    char name[30]="Suresh Techs";
    printf("Before: %s\n",name);
    strupr(name);
    printf("After: %s",name);
    return 0;
}
```

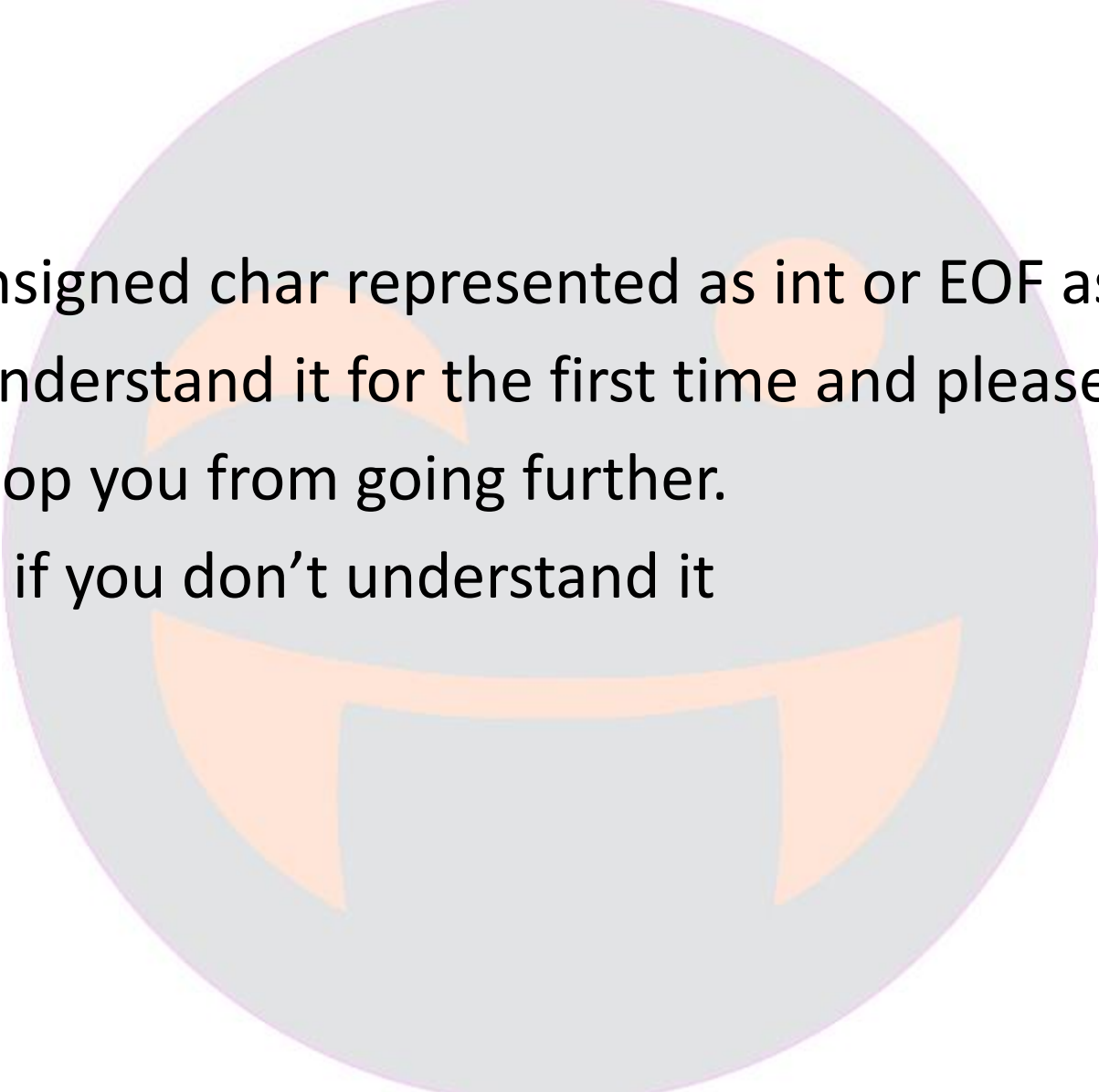
```
Before: Suresh Techs
After: SURESH TECHS
```

Punctuations

[] () { } . ->
++ -- & * + - ~ !
/ % << >> < > <= >= == != ^ | && ||
? : ; ...
= *= /= %= += -= <<= >>= &= ^= |=
, # ##
<: :> <% %> %: %::

Built in character functions - <ctype.h>

Function	Description
isalpha()	checks whether the character variable/constant contains alphabet or not
isdigit()	checks whether the character variable/ constant contains digit or not
isalnum()	checks whether the character variable/constant contains an alphabet or digit
ispunct()	checks whether the character variable/constant contains a punctuator or not. Punctuators are comma, semicolon etc
isspace()	checks whether the character variable/constant contains a space or not
isupper()	checks whether the character variable/constant contains an capital letter alphabet or not
islower()	checks whether the character variable/constant contains a lowercase alphabet or not
toupper()	converts lowercase alphabet into uppercase alphabet
tolower()	converts an uppercase alphabet into lowercase alphabet

- 
- Receives an unsigned char represented as int or EOF as argument.
 - You may not understand it for the first time and please don't worry.
 - This doesn't stop you from going further.
 - Move on even if you don't understand it

```
#include<stdio.h>
#include<ctype.h>
int main(){
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Alphabet: %d\n",isalpha('1'));
    printf("Is Alphabet: %d\n",isalpha('1d'));
    printf("Is Alphabet: %d\n",isalpha(','));
    printf("Is Alphabet: %d\n",isalpha('c'));
    printf("Is Alphabet: %d\n",isalpha('100b'));
    printf("Is Alphabet: %d\n",isalpha('100'));
    printf("Is Alphabet: %d\n",isalpha(100));

    return 0;
}
```

```
Is Alphabet: 0
Is Alphabet: 2
Is Alphabet: 0
Is Alphabet: 2
Is Alphabet: 2
Is Alphabet: 0
Is Alphabet: 2
```



```
#include<stdio.h>
#include<ctype.h>
int main() {
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Digit: %d\n",isdigit('1'));
    printf("Is Digit: %d\n",isdigit('11'));
    printf("Is Digit: %d\n",isdigit('1d'));
    printf("Is Digit: %d\n",isdigit(',')');
    printf("Is Digit: %d\n",isdigit('c'));
    printf("Is Digit: %d\n",isdigit('100b'));
    printf("Is Digit: %d\n",isdigit('100'));
    printf("Is Digit: %d\n",isdigit(100));

    return 0;
}
```

```
Is Digit: 1
Is Digit: 0
Is Digit: 0
Is Digit: 0
Is Digit: 0
Is Digit: 0
Is Digit: 0
Is Digit: 0
Is Digit: 0
```

```
#include<stdio.h>
#include<ctype.h>
int main(){
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Alphanumeric: %d\n",isalnum('1'));
    printf("Is Alphanumeric: %d\n",isalnum('11'));
    printf("Is Alphanumeric: %d\n",isalnum('1d'));
    printf("Is Alphanumeric: %d\n",isalnum(','));
    printf("Is Alphanumeric: %d\n",isalnum('c'));
    printf("Is Alphanumeric: %d\n",isalnum('100b'));
    printf("Is Alphanumeric: %d\n",isalnum('100'));
    printf("Is Alphanumeric: %d\n",isalnum(100));

    return 0;
}
```

```
Is Alphanumeric: 4
Is Alphanumeric: 4
Is Alphanumeric: 2
Is Alphanumeric: 0
Is Alphanumeric: 2
Is Alphanumeric: 2
Is Alphanumeric: 2
Is Alphanumeric: 4
Is Alphanumeric: 2
```

```
#include<stdio.h>
#include<ctype.h>
int main(){
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Space: %d\n",isspace('1'));
    printf("Is Space: %d\n",isspace(' 11'));
    printf("Is Space: %d\n",isspace('1d '));
    printf("Is Space: %d\n",isspace(' '));
    printf("Is Space: %d\n",isspace(', '));
    printf("Is Space: %d\n",isspace('c'));
    printf("Is Space: %d\n",isspace('100b'));
    printf("Is Space: %d\n",isspace('100'));
    printf("Is Space: %d\n",isspace(100));

    return 0;
}
```

```
Is Alphanumeric: 0
Is Alphanumeric: 0
Is Alphanumeric: 8
Is Alphanumeric: 8
Is Alphanumeric: 0
Is Alphanumeric: 0
Is Alphanumeric: 0
Is Alphanumeric: 0
Is Alphanumeric: 0
Is Alphanumeric: 0
```

```
#include<stdio.h>
#include<ctype.h>
int main() {
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Punctuation: %d\n",ispunct('['));
    printf("Is Punctuation: %d\n",ispunct(' 11'));
    printf("Is Punctuation: %d\n",ispunct('1d '));
    printf("Is Punctuation: %d\n",ispunct(' '));
    printf("Is Punctuation: %d\n",ispunct(',')');
    printf("Is Punctuation: %d\n",ispunct('\\'));
    printf("Is Punctuation: %d\n",ispunct(']b'));
    printf("Is Punctuation: %d\n",ispunct('100'));
    printf("Is Punctuation: %d\n",ispunct(100));

    return 0;
}
```

```
Is Punctuation: 16
Is Punctuation: 0
Is Punctuation: 0
Is Punctuation: 0
Is Punctuation: 16
Is Punctuation: 16
Is Punctuation: 0
Is Punctuation: 0
Is Punctuation: 0
```

```
#include<stdio.h>
#include<ctype.h>
int main(){
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Upper: %d\n",isupper('D'));
    printf("Is Upper: %d\n",isupper('11'));
    printf("Is Upper: %d\n",isupper('d'));
    printf("Is Upper: %d\n",isupper('sT'));
    printf("Is Upper: %d\n",isupper('cc'));
    printf("Is Upper: %d\n",isupper('k'));
    printf("Is Upper: %d\n",isupper('RRR'));
    printf("Is Upper: %d\n",isupper('100'));
    printf("Is Upper: %d\n",isupper(100));

    return 0;
}
```

```
Is Upper: 1
Is Upper: 0
Is Upper: 0
Is Upper: 1
Is Upper: 0
Is Upper: 0
Is Upper: 1
Is Upper: 0
Is Upper: 0
```

```
#include<stdio.h>
#include<ctype.h>
int main() {
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("Is Lower: %d\n",islower('D'));
    printf("Is Lower: %d\n",islower('11'));
    printf("Is Lower: %d\n",islower('d'));
    printf("Is Lower: %d\n",islower('sT'));
    printf("Is Lower: %d\n",islower('cc'));
    printf("Is Lower: %d\n",islower('k'));
    printf("Is Lower: %d\n",islower('RRR'));
    printf("Is Lower: %d\n",islower('100'));
    printf("Is Lower: %d\n",islower(100));

    return 0;
}
```

```
Is Lower: 0
Is Lower: 0
Is Lower: 2
Is Lower: 0
Is Lower: 2
Is Lower: 2
Is Lower: 0
Is Lower: 0
Is Lower: 2
```

```
#include<stdio.h>
#include<ctype.h>
int main(){
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("To Upper: %c\n",toupper('D'));
    printf("To Upper: %c\n",toupper('11'));
    printf("To Upper: %c\n",toupper('d'));
    printf("To Upper: %c\n",toupper('st'));
    printf("To Upper: %c\n",toupper('cc'));
    printf("To Upper: %c\n",toupper('k'));
    printf("To Upper: %c\n",toupper(' '));
    printf("To Upper: %c\n",toupper('100'));
    printf("To Upper: %c\n",toupper(100));

    return 0;
}
```

```
To Upper: D
To Upper: 1
To Upper: D
To Upper: t
To Upper: c
To Upper: K
To Upper: 
To Upper: 0
To Upper: D
```

```
#include<stdio.h>
#include<ctype.h>
int main() {
    for(int i = 0;i<256;i++){
        printf("ASCII: %d = %c\n",i,i);
    }
    printf("To Lower: %c\n",tolower('D'));
    printf("To Lower: %c\n",tolower('11'));
    printf("To Lower: %c\n",tolower('d'));
    printf("To Lower: %c\n",tolower('st'));
    printf("To Lower: %c\n",tolower('cc'));
    printf("To Lower: %c\n",tolower('k'));
    printf("To Lower: %c\n",tolower(' '));
    printf("To Lower: %c\n",tolower('100'));
    printf("To Lower: %c\n",tolower(100));

    return 0;
}
```

```
To Lower: d
To Lower: 1
To Lower: d
To Lower: t
To Lower: c
To Lower: k
To Lower: 
To Lower:  
To Lower: d
```


What next?

- Structures

