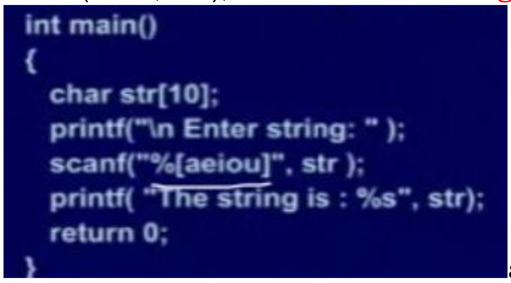
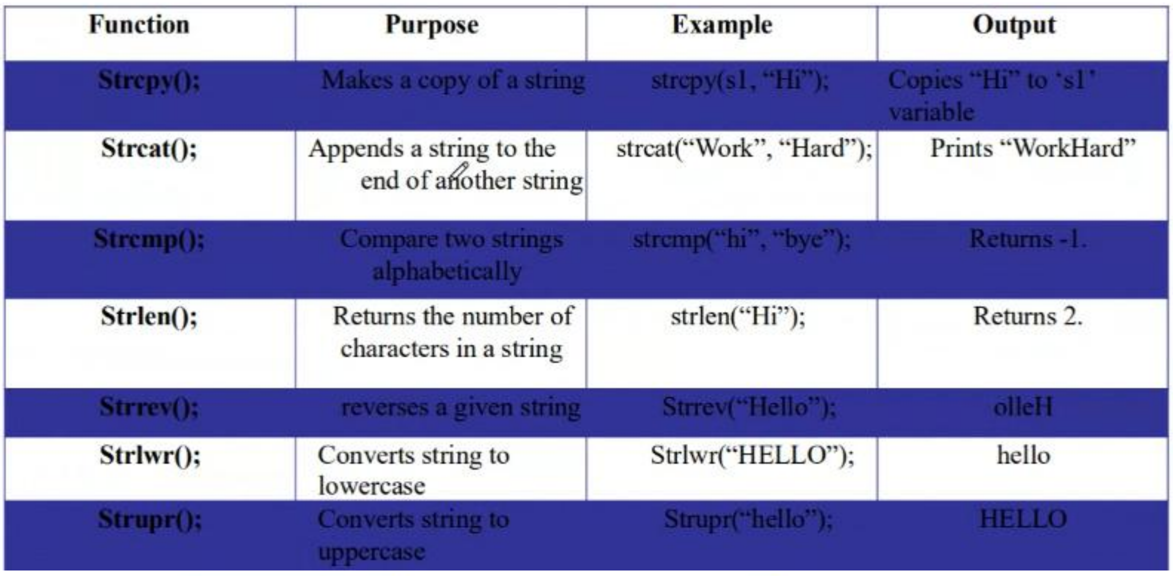
Unit IV

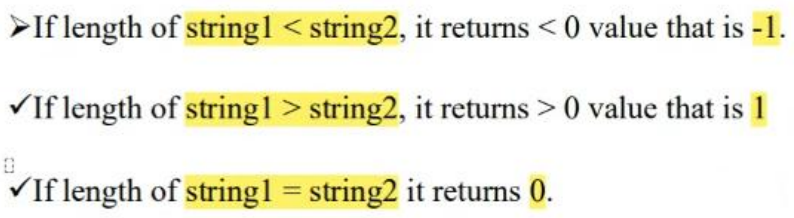
STRINGS:

- Collection of individual array elements.
- Enclosed within double quotes.
- Always terminated by null character "\0" [imp for string length]
- syntax: char str[size]
- char city[] → compile time error
- operations on strings: #include <string.h>
- string initializations:
    - char str[50] = "Hello world"
    - char greeting[] = {"h","e","l","l","o"}
- scanf("%s", str); #NO & like in integers.
- 

```
int main()
{
  char str[10];
  printf("\n Enter string: " );
  scanf("%[aeiou]", str );
  printf( "The string is : %s", str);
  return 0;
}
```

  accepts only i/p specified. Here, aeiou
- If %[^aeiou] accepts all i/p apart from specified.
- ASCII for A -Z (65 to 91) and a -z (97 to 123)
- Lower case to upper case: subtracts 32 from character.
- 

Function	Purpose	Example	Output
Strcpy();	Makes a copy of a string	strcpy(s1, "Hi");	Copies "Hi" to 's1' variable
Strcat();	Appends a string to the end of another string	strcat("Work", "Hard");	Prints "WorkHard"
Strcmp();	Compare two strings alphabetically	strcmp("hi", "bye");	Returns -1.
Strlen();	Returns the number of characters in a string	strlen("Hi");	Returns 2.
Strrev();	reverses a given string	Strrev("Hello");	olleH
Strlwr();	Converts string to lowercase	Strlwr("HELLO");	hello
Strupr();	Converts string to uppercase	Strupr("hello");	HELLO

- 

➢If length of string1 < string2, it returns < 0 value that is -1.

✓If length of string1 > string2, it returns > 0 value that is 1

✓If length of string1 = string2 it returns 0.

strcmp

FUNCTIONS:

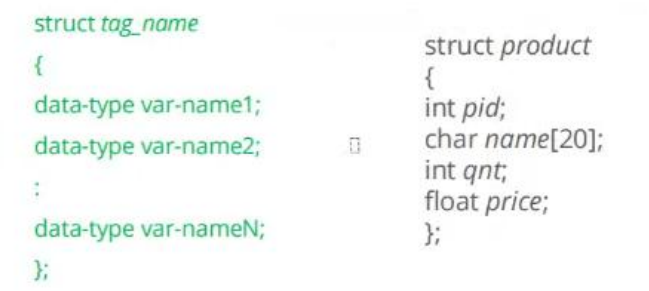
- It is a block of code that performs a specific task.

- It has a name and is reusable in different parts of the program.
- It also optionally returns a value to the calling program.
- Types of functions:
    - void function(void)
    - void function(int)
    - int function(void)
    - int function(int)

Unit V

STRUCTURES:

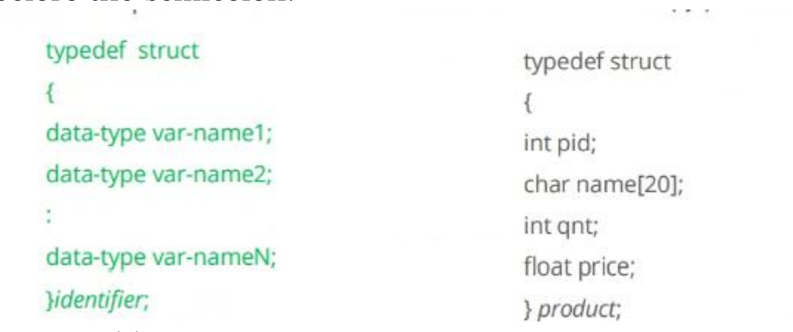
- Array: a user defined type stores data element of same datatype
- Structure: user defined type that can hold a collection of elements of different datatypes.
- Declaring:
  ```
  struct student{
  char name[20];
  char usn[10];
  int courses;
  float marks1, marks2, marks3;
  } S1, S2, S3;
  ```
- Two ways of declaring or defining a structure:
    - Tagged: starts with the keyword struct followed by tag name



```
struct tag_name
{
data-type var-name1;
data-type var-name2;
:
data-type var-nameN;
};
```

```
struct product
{
int pid;
char name[20];
int qnt;
float price;
};
```

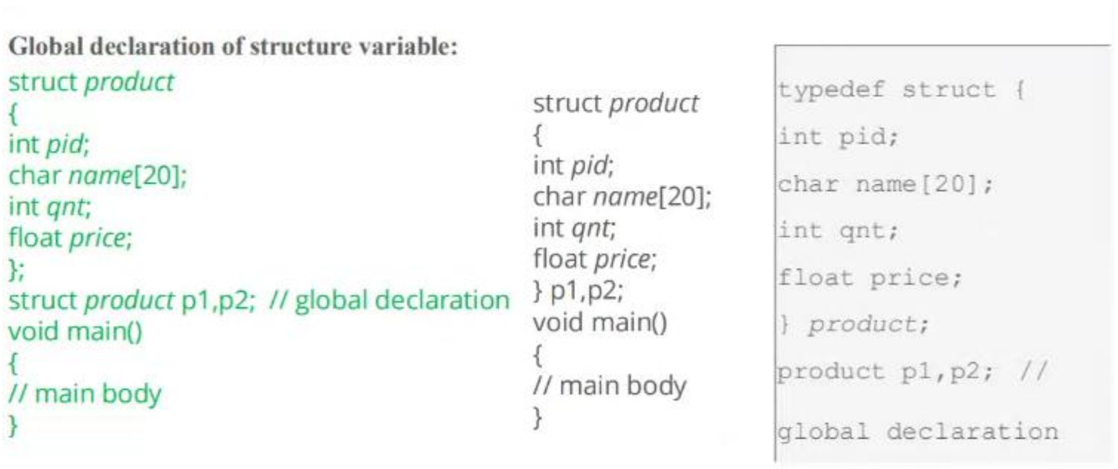
    - Typedef: required an identifier ar the end of the structure block and before the semicolon.



```
typedef struct
{
data-type var-name1;
data-type var-name2;
:
data-type var-nameN;
}identifier;
```

```
typedef struct
{
int pid;
char name[20];
int qnt;
float price;
} product;
```

- Structure variable syntax: struct <struct_name> var_name

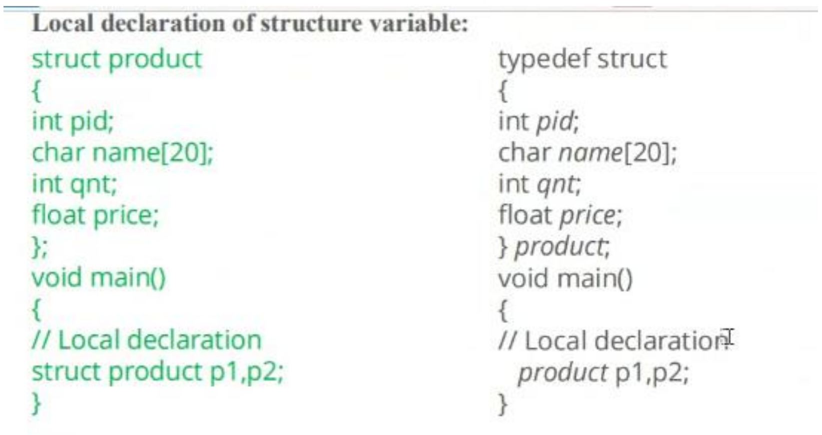


Global declaration of structure variable:

```
struct product
{
int pid;
char name[20];
int qnt;
float price;
};
struct product p1,p2;  // global declaration
void main()
{
// main body
}
```

```
struct product
{
int pid;
char name[20];
int qnt;
float price;
} p1,p2;
void main()
{
// main body
}
```

```
typedef struct {
int pid;
char name[20];
int qnt;
float price;
} product;
product p1,p2;  //
global declaration
```

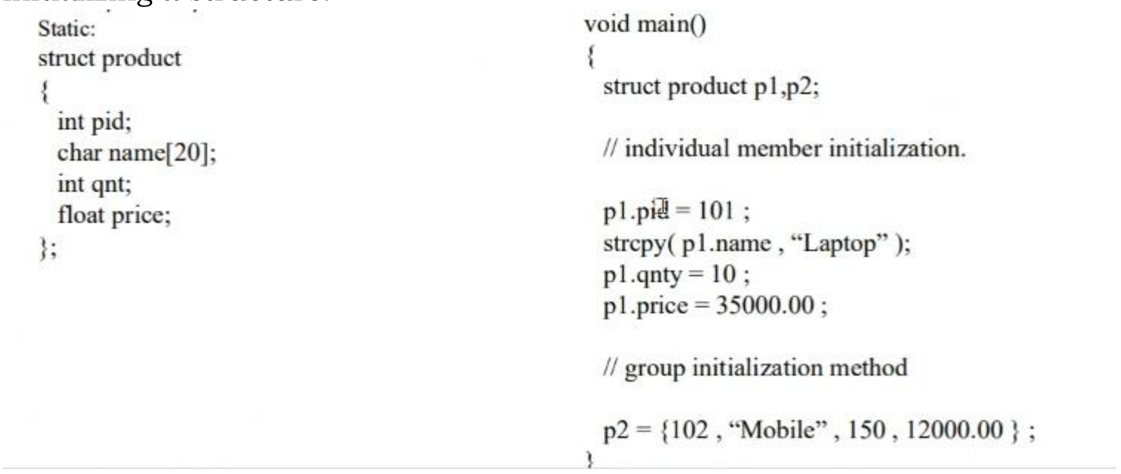
- 

Local declaration of structure variable:

```
struct product
{
int pid;
char name[20];
int qnt;
float price;
};
void main()
{
// Local declaration
struct product p1,p2;
}
```

```
typedef struct
{
int pid;
char name[20];
int qnt;
float price;
} product;
void main()
{
// Local declaration
  product p1,p2;
}
```

-
- Each structure member is allocated separate memory area.
- To access individual structure member: the structure member operator(.) aka direct selection operator
  - syntax: struct_var.member_name
- initializing a structure:



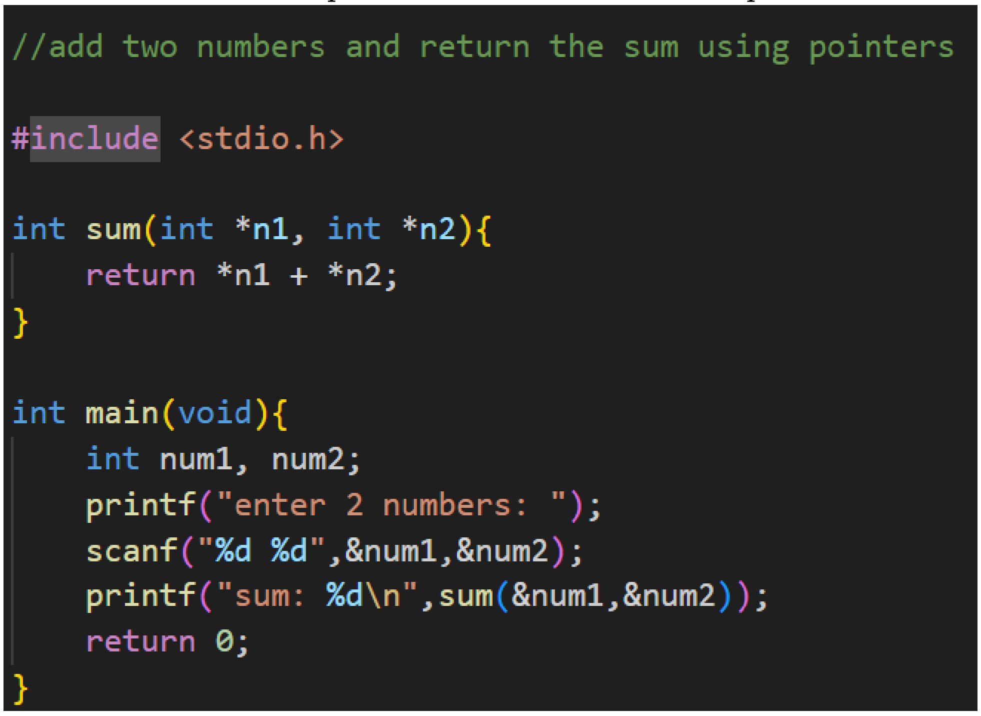
Static:

```
struct product
{
  int pid;
  char name[20];
  int qnt;
  float price;
};
```

```
void main()
{
  struct product p1,p2;

  // individual member initialization.

  p1.pid = 101 ;
  strcpy( p1.name , "Laptop" );
  p1.qnty = 10 ;
  p1.price = 35000.00 ;

  // group initialization method

  p2 = {102 , "Mobile" , 150 , 12000.00 } ;
}
```

POINTERS

- provides a way of accessing a variable without referring to the variable directly.
- The mechanism used for this is the address of the variable

- The prog stmt can refer to a variable indirectly using the address of the variable.
- Pointer variable stores the memory address of the variable
- Pointer holds address rather than a value thus it has 2 parts:
  - The pointer itself holds the address
  - The address points to a value
- Returns more than one value from the function indirectly.
- The pointer operator is * aka address operator.
- The value at address operator is called indirection operator.
- 

```c
//add two numbers and return the sum using pointers

#include <stdio.h>

int sum(int *n1, int *n2){
    return *n1 + *n2;
}

int main(void){
    int num1, num2;
    printf("enter 2 numbers: ");
    scanf("%d %d",&num1,&num2);
    printf("sum: %d\n",sum(&num1,&num2));
    return 0;
}
```