**POINTERS**

**1. Array of pointers to store string.**

e.g.

#include <stdio.h>

const int MAX = 4;

int main () {

char \*names[] = {

"Zara Ali",

"Hina Ali",

"Nuha Ali",

"Sara Ali"

};

int i = 0;

for ( i = 0; i < MAX; i++) {

printf("Value of names[%d] = %s\n", i, names[i] );

}

return 0;

}

**2. Return Pointer from function.**

It is not a good idea to return the **address of a local variable** outside the function, so you would have to define the local variable as **static** variable.

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int \* getRandom( ) {

static int r[10]; // this variable is declared as static as we are returing its address outside the //function

int i;

for ( i = 0; i < 10; ++i) {

r[i] = rand();

printf("%d\n", r[i] );

}

return r;

}

int main () {

int \*p;

int i;

p = getRandom();

for ( i = 0; i < 10; i++ ) {

printf("\*(p + [%d]) : %d\n", i, \*(p + i) );

}

return 0;

}

**STRING**

**Some inbuilt functions:**

**1.strchr(s1,ch)**

**char \* strchr ( const char \*, int );**

Returns a pointer to the first occurrence of character ch in string s1.

#include<stdio.h>

#include<string.h>

void main(){

char str[] ="i am gonna strive hard for the next 6 months.";

char ch = 'm';

char \*ptr;

ptr = strchr(str, ch);

int k=1;

while(ptr!=NULL){

printf("occurance: %d", k);

printf("\nthe position of the occurance of the word |%c| is %ld", ch, ptr-str+1);

printf("\nthe string after the charecter |%c| is |%s|\n\n\n", ch, ptr);

ptr= strchr(ptr+1, ch);

k++;

}}

**2.strstr(s1,s2).**

Returns a pointer to the first occurrence of string s2 in string s1.

**3. strcpy()**

**4. strcat()**

**5. strcom(**)

**STRUCTURE:**

**struct [structure tag] {**

**member definition;**

**member definition;**

**...**

**member definition;**

**} [one or more structure variables];**

**STRUCTURE as function argument:**

#include<stdio.h>

#include<string.h>

struct kaka\_pg {

char name[10];

char institute[10];

char course[10];

} Ashok;

void func(struct kaka\_pg uncle);

void main(){

strcpy(Ashok.name, "vivek");

strcpy(Ashok.institute, "vector");

strcpy(Ashok.course, "embedded system");

func(Ashok);

}

void func(struct kaka\_pg uncle){

printf("name of the student:%s\n", uncle.name);

printf("name of the institute:%s\n", uncle.institute);

printf("name of the course:%s\n", uncle.course);

}

**Pointer to structure:**

**struct [name of structure] \*structure pointer;**

we can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the '&'; operator before the structure's name as follows--

**struct\_pointer = &structure variable;**

#include<stdio.h>

#include<string.h>

struct kaka\_pg {

char name[10];

char institute[10];

char course[16];

} Ashok;

void func(struct kaka\_pg \*uncle);

void main(){

strcpy(Ashok.name, "vivek");

strcpy(Ashok.institute, "vector");

strcpy(Ashok.course, "embedded system");

func(&Ashok);

}

void func(struct kaka\_pg \*uncle){

printf("name of the student:%s\n", uncle->name);

printf("name of the institute:%s\n", uncle->institute);

printf("name of the course:%s\n", uncle->course);

}

**Dynamic memory allocation:**

**1. malloc :** stands for memory allocation.

The function **malloc()** reserves a block of memory of specified size and return a pointer of type **void**, which can be casted into a pointer of any form.

syntex: ptr =(cast\_type \*) malloc (size);

mallocl() does not initialize the memory location with any value. i.e. the memory allocate with the hepl of malloc() contains garbage value.