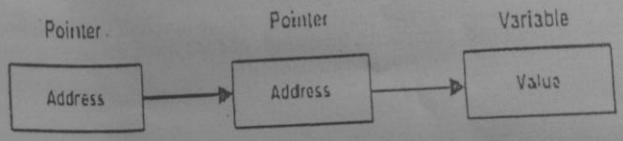
POINTERS

Q. What is pointer? How pointers and arrays are closely related? Briefly discuss with example?Marks: 3
Exam:ACCE-2014,13,ICE-2013,15 CSE-2011
Q.Define mechanism of pointer? Exam:
Ans:

A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before you can use it to store any variable address.
Or
A Pointer is just an address of the data stored in memory.
Or. , A pointer variable (or just "pointer") is a reference to another variable. Its can contain the memory address of any variable type:

Example:

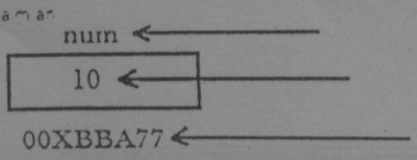


Reference operator(&)
If *var* is a variable then, &var is the address in memory.

```
Consider the below code:
int var = 1;
int main()
{
  int num = 10;
  printf("Value of var is: %d", num);
  printf("Address of var is: %u", &num);
  return 0;
}
Output:
Value of var is: 10
Address of var is: 00XBBA77
```

n the above code we have two variables of type integer. One is num and another one is var. The value of num is 10 and this value should be stored somewhere in the memory, rite? Yes compiler allocates a memory space for each of the variable which is known as address of the variable and it should be in hexadecimal format.

So let's say computer assigned a location 00XBBA77 for variable num, which means whatever value we would be assigning to num should be stored at location: 00XBBA77.



& Operator – "Address of" Operator
As you can see that in above example, I have used & operator to find out the address of variable name. & operator is also known as "Address of" Operator.
printf("Address of var is: %u", &num);

- **Operator – "Value at Address" Operator**
- Operator is also known as Value at address operator.

As we discussed above, we can store the address of a variable in another variable which is known as pointer.

Q.How to declare a pointer?

Ans:

The general form of a pointer variable declaration is:

```
type *var-name;
int *ip;          pointer to an integer
double *dp;    Pointer to a variable of data type double to a double
float *fp;      pointer to a float
char *ch        pointer to a character
```

The above are the few examples of pointer declarations. If you need a pointer to store the address of **integer variable then the data type of the pointer should be int.**

Lets take the same example with a pointer variable –

```
int main()
{
  int *p
  int var = 10;
  p= &var;
printf("Value of var is: %d", var);
  printf("Address of var is:
%u", p);
  return 0;
}
```

p	var
00XBBA77	10
77221111	00XBBA77

Q.Write a program to demonstrate, handling of pointers in C ? Exam:

Ans:

```
#include <stdio.h>
int main(){
  int* pc;
  int c;
  c=22;
  printf("Address of c:%d ",&c);
  printf("Value of c:%d",c);
  pc=&c;
  printf("Address of pointer pc:%d",pc);
  printf("Content of pointer pc:%d ",*pc);
  c=11;
  printf("Address of pointer pc:%d\n",pc);
  printf("Content of pointer pc:%d\n",*pc);
  *pc=2;
  printf("Address of c:%d\n",&c);
  printf("Value of c:%d\n",c);
  return 0;
}
```

Output

```
Address of c: 2686784
Value of c: 22
```

Address of pointer pc: 2686784
Content of pointer pc: 22

Address of pointer pc: 2686784
Content of pointer pc: 11

Address of c: 2686784
Value of c: 2

Q.Explain Above pointer example's programs?

Ans:

Explanation of program and figure

1. Code int* pc; creates a pointer *pc* and a code int c; creates normal variable *c*. Pointer *pc* points to some address and that address has garbage value. Similarly, variable *c* also has garbage value at this point.
2. Code c=22; makes the value of c equal to 22, i.e.,22 is stored in the memory location of variable *c*.
3. Code pc=&c; makes pointer, point to address of c. Note that, &c is the address of variable *c* (because *c* is normal variable) and *pc* is the address of *pc* (because pc is the pointer variable). Since the address of pc and address of c is same, *pc will be equal to the value of *c*.
4. Code c=11; makes the value of *c*, 11. Since, pointer *pc* is pointing to address of *c*. Value inside address *pc* will also be 11.
5. Code *pc=2; change the contents of the memory location pointed by pointer *pc* to change to 2. Since address of pointer *pc* is same as address of *c*, value of *c* also changes to 2.

Q.Suppose, the programmer want pointer pc to point to the address of c .
int c, *pc; pc=c; *pc=&c; Explain it.

Ans;

```
pc=c;  /* pc is address whereas, c is not an address. */
*pc=&c; /* &c is address whereas, *pc is not an address. */
```

Q. How to declare a pointer to Pointer (Or Double Pointer) in C?

Ans:

```
int **pr;
```

There should be two *s before double pointer variable

Consider the below figure and program to understand this concept better – Diagram

pr1	pr2	num
66X123X1	XX771230	123
XX661111	66X123X1	XX771230

As per the figure, pr2 is a pointer for num (as pr2 is having address of variable num), similarly pr1 is a pointer for another pointer pr1 (as pr1 is having the address of pointer pr2). A pointer which points to another pointer is known as double pointer. In this example pr1 is a double pointer.

Values from above diagram –

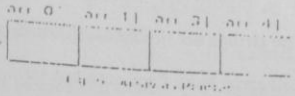
Variable num has address: XX771230
Address of Pointer pr1 is: XX661111
Address of Pointer pr2 is: 66X123X1

February 1, 2016

Q. Describe the arrays of pointers?
Or How pointers and arrays are closely related? Briefly discuss with example? Marks:3 Exam-ACCE-2014,ICE-2013,CSE-APPE
Ans:

Relation between Arrays and Pointers:

Consider and array:
int arr[4];



In arrays of C programming, name of the array points to the first element of an array.

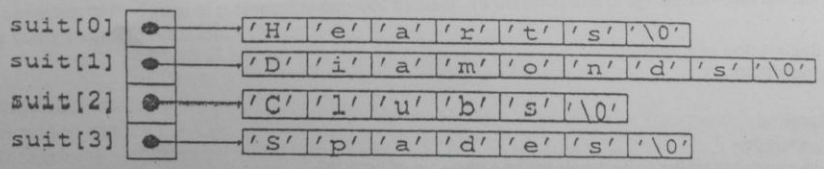
always

Here, address of first element of an array is &arr[0].
Also, *arr* represents the address of the pointer where it is pointing. Hence, &arr[0] is equivalent to *arr*.
Also, value inside the address &arr[0] and address *arr* are equal. Value in address &arr[0] is arr[0] and value in address *arr* is *arr.
Hence, arr[0] is equivalent to *arr.
Similarly,
&a[1] is equivalent to (a+1) AND, a[1] is equivalent to *(a+1).
&a[2] is equivalent to (a+2) AND, a[2] is equivalent to *(a+2).
&a[3] is equivalent to (a+1) AND, a[3] is equivalent to *(a+3).
.
.
&a[i] is equivalent to (a+i) AND, a[i] is equivalent to *(a+i).
In C, you can declare an array and can use pointer to alter the data of an array.

Program to find the sum of six numbers with arrays and pointers.

```
#include <stdio.h>
int main(){
 int i,class[6],sum=0;
 printf("Enter 6 numbers:\n");
 for(i=0;i<6;++i){
   scanf("%d",(class+i)); // (class+i) is equivalent to &class[i]
   sum += *(class+i);  // *(class+i) is equivalent to class[i]
 }
 printf("Sum=%d",sum);
 return 0;
}
```

Q.Disscuss Array as a pointer?
Ans:
Arrays can contain pointers
For example: an array of strings
char *suit[4] = { "Hearts", "Diamonds",
"Clubs", "Spades" };



- suit are pointers to the first character
- char * – each element of suit is a pointer to a char

- The strings are not actually stored in the array suit, only pointers to the strings are stored
- suit array has a fixed size, but strings can be of any size

Q.How to use Pointers?
Ans:
There are few important operations
which we will do with the help of pointers very frequently.
(a) we define a pointer variable
(b) assign the address of a variable to a pointer and
(c) finally access the value at the address available in the pointer variable.

Q.Why use Pointer or Advantage of pointer? Marks: Exam:ACCE-2013 ICE,CSE,APPE
Ans:

- They allow you to refer to large data structures in a compact way
- They facilitate sharing between different parts of programs
- They make it possible to get new memory dynamically as your program is running
- They make it easy to represent relationships among data items.

Q. Distinguish between malloc() & calloc() memory allocation.
Both allocates memory from heap area/dynamic memory. By default calloc fills the allocated memory with 0's.

Q.What is keyword auto for?
By default every local variable of the function is automatic (auto). In the below function both the variables 'i' and 'j' are automatic variables.

```
void f() {
  int i;
  auto int j;
}
```

NOTE – A global variable can't be an automatic variable.

Q.Describe How can you declare pointer variable give with an example? Exam:ACCE
To declare a pointer variable, we must do two things

- Use the "*" (star) character to indicate that the variable being defined is a pointer type.
- Indicate the type of variable to which the pointer will point (the pointee).

General declaration of a pointer
type *nameOfPointer;

The declaration
int *intPtr;
defines the variable intPtr to be a pointer to a variable of type int.

Pointer Examples:

```
int x = 1, y = 2, z[10];
int *ip;          ip is a pointer to an int
ip = &x;          ip points to (contains the memory address of) x
y = *ip;          y is now 1, indirectly copied from x using ip
*ip = 0;          x is now 0
ip = &z[5];       Ip now points to z[5]
```

Q.Defien dynamic memory allocation. Why it is required? Marks: 2 Exam:ACCE-2014
Ans:
C Dynamic Memory Allocation: malloc(), calloc(), free() & realloc()

February 1, 2016

.The size of array you have declared initially can be sometimes insufficient and sometimes more than required. Dynamic memory allocation allows a program to obtain more memory space, while running or to release space when no space is required.

Although, C language inherently does not has any technique to allocated memory dynamically, there are 4 library functions under "stdlib.h" for dynamic memory allocation.

Function	Use of Function
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory
free()	dellocate the previously allocated space
realloc()	Change the size of previously allocated space

malloc()

The name 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 pointer of any form.

Syntax of malloc()

```
ptr=(cast-type*)malloc(byte-size)
```

Here, *ptr* is pointer of cast-type. The malloc() function returns a pointer to an area of memory with size of byte size. If the space is insufficient, allocation fails and returns NULL pointer.

```
ptr=(int*)malloc(100*sizeof(int));
```

This statement will allocate either 200 or 400 according to size of int 2 or 4 bytes respectively and the pointer points to the address of first byte of memory.

calloc()

The name calloc stands for "contiguous allocation". The only difference between malloc() and calloc() is that, malloc() allocates single block of memory whereas calloc() allocates multiple blocks of memory each of same size and sets all bytes to zero.

Syntax of calloc()

```
ptr=(cast-type*)calloc(n,element-size);
```

This statement will allocate contiguous space in memory for an array of *n* elements. For example:

```
ptr=(float*)calloc(25,sizeof(float));
```

This statement allocates contiguous space in memory for an array of 25 elements each of size of float, i.e, 4 bytes.

free()

Dynamically allocated memory with either calloc() or malloc() does not get return on its own. The programmer must use free() explicitly to release space.

syntax of free()

```
free(ptr);
```

This statement cause the space in memory pointer by ptr to be deallocated.

Examples of malloc():

Q.Write a C program to find sum of n elements entered by user. To perform this program, allocate memory dynamically using malloc() function.

Ans:

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  int n,i,*ptr,sum=0;
  printf("Enter number of elements: ");
  scanf("%d",&n);
  ptr=(int*)malloc(n*sizeof(int)); //memory allocated using malloc
  if(ptr==NULL)
```

```
    {
      printf("Error! memory not allocated.");
      exit(0);
    }
    printf("Enter elements of array: ");
    for(i=0;i<n;++i)
    {
      scanf("%d",ptr+i);
      sum+=*(ptr+i);
    }
    printf("Sum=%d",sum);
    free(ptr);
    return 0;
}
```

Why required:

We need to use dynamic memory when:

- we cannot determine the maximum amount of memory to use at compile time;
- we want to allocate a *very* large object;
- we want to build data structures (containers) without a fixed upper size;

Q.Describe Use of Pointer ?

Ans: use of pointer:

- A primitive (int, char, float)
- An array
- A struct or union
- Dynamically allocated memory
- Another pointer
- A function

Q. Passing pointers to functions in C with example?

Ans: Pointers can also be passed as an argument to a function. Below example will help you understand how to do it.

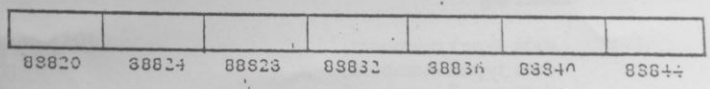
```
int salaryhike(int *var, int hike)
{
    *var = *var+hike;
    return *var;
}
int main()
{
    int sal=0;
    int hike=0;
    printf("Enter the employee current salary:");
    scanf("%d", &sal);
    printf("Enter hike amount:");
    scanf("%d", &hike);
    int op = salaryhike (&sal, hike);
    printf("Final salary: %d", op);
 return 0;
}
```

Output:
Enter the employee current salary: 35000
Enter hike amount: 3500
Final salary: 38500

Q. Pointer to an array in C programming with example?
Or
How pointer use as an array give example?
Ans:

```
include <stdio.h>
int main()
{
 int val[7] = { 11, 22, 33, 44, 55, 66, 77 } ;
 for ( int i = 0 ; i <= 6 ; i++ )
  {
    printf("val[%d]: value is %d and address is %u", i, val[i], &val[i]);
  }
 return 0;
}
```

Output:
val[0]: value is 11 and address is 88820
val[1]: value is 22 and address is 88824
val[2]: value is 33 and address is 88828
val[3]: value is 44 and address is 88832
val[4]: value is 55 and address is 88836
val[5]: value is 66 and address is 88840
val[6]: value is 77 and address is 88844



In the above example I have used &val[i] to get the address of ith element of the array. We can also use a pointer variable instead of it.
Consider the same above example with pointers –

```
#include <stdio.h>
int main()
{

 int *p;
  int val[7] = { 11, 22, 33, 44, 55, 66, 77 } ;
  p = &val[0];
 for ( int i = 0 ; i <= 6 ; i++ )
 {
  printf("val[%d]: value is %d and address is %u", i, *p, p);
  p++;
 }
 return 0;
}
```

The above program would result the same output, that our first program returned.