Unit-4 Pointers & Array

{C}
Programming

Need of Array Variable

- Suppose we need to store rollno of the student in the integer variable.

Declaration

```
int rollno;
```

- Now we need to store rollno of 100 students.

Declaration

```
int rollno101, rollno102, rollno103, rollno104...;
```

- This is not appropriate to declare these many integer variables.

e.g. 100 integer variables for rollno.

- Solution to declare and store multiple variables of similar type is an array.
- An array is a variable that can store multiple values.

Definition: Array

- An array is a fixed size sequential collection of elements of same data type grouped under single variable name.

```
int rollno[100];
```

[0]	[1]	[2]	…	[99]

Fixed Size	Sequential	Same Data type	Single Name
Here, the size of an array is 100 (fixed) to store rollno	It is indexed to 0 to 99 in sequence	All the elements (0-99) will be integer variables	All the elements (0-99) will be referred as a common name rollno

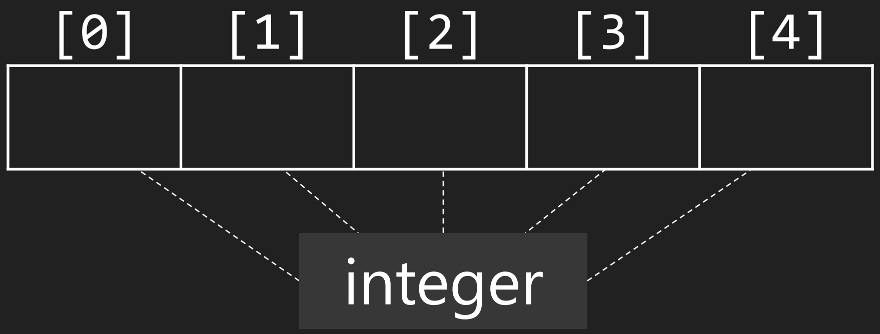
Declaring an array

Syntax

```
data-type variable-name[size];
```

Integer Array

```
int mark[5];
```

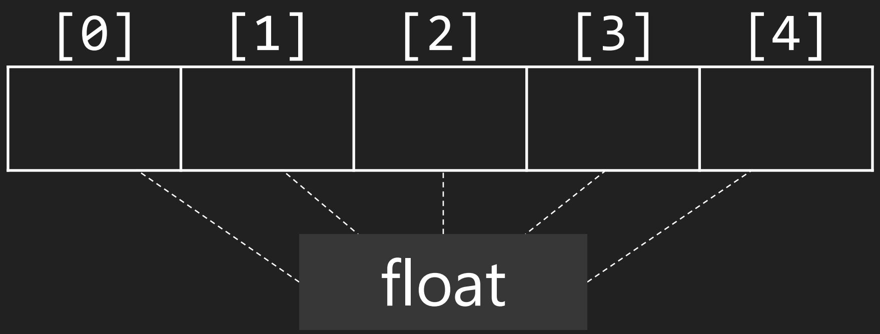


[0] [1] [2] [3] [4]

integer

Float Array

```
float avg[5];
```



[0] [1] [2] [3] [4]

float

- By default array index starts with 0.
- If we declare an array of size 5 then its index ranges from 0 to 4.
- First element will be store at mark[0] and last element will be stored at mark[4] not mark[5].
- Like integer and float array we can declare array of type char.

Initialing and Accessing an Array

Declaring, initializing and accessing single integer variable

```c
int mark=90;      //variable mark is initialized with value 90
printf("%d",mark); //mark value printed
```

Declaring, initializing and accessing integer array variable

```c
int mark[5]={85,75,76,55,45}; //mark is initialized with 5 values
printf("%d",mark[0]); //prints 85
printf("%d",mark[1]); //prints 75
printf("%d",mark[2]); //prints 65
printf("%d",mark[3]); //prints 55
printf("%d",mark[4]); //prints 45
```

	[0]	[1]	[2]	[3]	[4]
mark[5]	85	75	65	55	45

Read(Scan) Array Elements

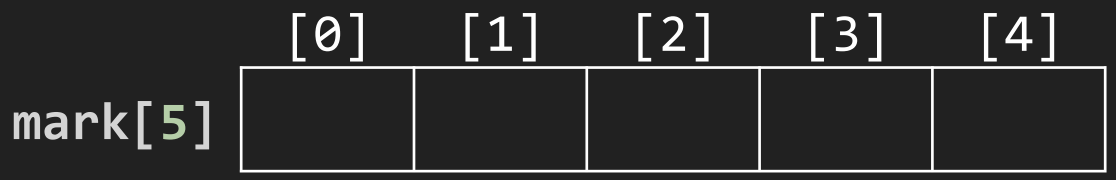
Reading array without loop

```c
void main()
{
  int mark[5];
  printf("Enter array element=");
  scanf("%d",&mark[0]);
  printf("Enter array element=");
  scanf("%d",&mark[1]);
  printf("Enter array element=");
  scanf("%d",&mark[2]);
  printf("Enter array element=");
  scanf("%d",&mark[3]);
  printf("Enter array element=");
  scanf("%d",&mark[4]);

  printf("%d",mark[0]);
  printf("%d",mark[1]);
  printf("%d",mark[2]);
  printf("%d",mark[3]);
  printf("%d",mark[4]);
}
```

Reading array using loop

```c
void main()
{
  int mark[5],i;
  for(i=0;i<5;i++)
  {
   printf("Enter array element=");
   scanf("%d",&mark[i]);
  }
  for(i=0;i<5;i++)
  {
   printf("%d",mark[i]);
  }
}
```



Develop a program to count number of positive or negative number from an array of 10 numbers.

Program

```c
void main(){
    int num[10],i,pos,neg;
    pos = 0;
    neg = 0;
    for(i=0;i<10;i++)
    {
        printf("Enter array element=");
        scanf("%d",&num[i]);
    }
    for(i=0;i<10;i++)
    {
        if(num[i]>0)
            pos=pos+1;
        else
            neg=neg+1;
    }
    printf("Positive=%d,Negative=%d",pos,neg);
}
```

Output

```
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
Enter array element=-1
Enter array element=-2
Enter array element=3
Enter array element=4
Enter array element=5
Positive=8,Negative=2
```

Develop a program to read n numbers in an array and print them in reverse order.

Program

```c
void main()
{
    int num[100],n,i;
    printf("Enter number of array elements=");
    scanf("%d",&n);
//loop will scan n elements only
    for(i=0;i<n;i++)
    {
        printf("Enter array element=");
        scanf("%d",&num[i]);
    }
//negative loop to print array in reverse order
    for(i=n-1;i>=0;i--)
    {
        printf("%d\n",num[i]);
    }
}
```

Output

```
Enter number of array
elements=5
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
5
4
3
2
1
```

Practice Programs

1) Develop a program to calculate sum of n array elements in C.
2) Develop a program to calculate average of n array elements in C.
3) Develop a program to find largest array element in C.
4) Develop a program to print sum of second and second last element of an array.
5) Develop a program to copy array elements to another array.
6) Develop a program to count odd and even elements of an array.

Multi Dimensional Array

Declaring 2 Dimensional Array

Syntax

```
data-type variable-name[x][y];
```

Declaration

```
int data[3][3]; //This array can hold 9 elements
```

```
int data[3][3];
```

	Column-0	Column-1	Column-2
Row-0	data[0][0]	data[0][1]	data[0][2]
Row-1	data[1][0]	data[1][1]	data[1][2]
Row-2	data[2][0]	data[2][1]	data[2][2]

- A two dimensional array can be seen as a table with 'x' rows and 'y' columns.
- The row number ranges from `0` to `(x-1)` and column number ranges from `0` to `(y-1)`.

Initialing and Accessing a 2D Array: Example-1

Program

```c
int data[3][3] = {
{1,2,3}, //row 0 with 3 elements
{4,5,6}, //row 1 with 3 elements
{7,8,9}  //row 2 with 3 elements
    };
printf("%d",data[0][0]); //1
printf("%d",data[0][1]); //2
printf("%d\n",data[0][2]); //3

printf("%d",data[1][0]); //4
printf("%d",data[1][1]);  //5
printf("%d\n",data[1][2]);  //6

printf("%d",data[2][0]);//7
printf("%d",data[2][1]); //8
printf("%d",data[2][2]); //9
```

```c
// data[3][3] can be initialized like this also
int data[3][3]={{1,2,3},{4,5,6},{7,8,9}};
```

	Column-0	Column-1	Column-2
Row-0	1	2	3
Row-1	4	5	6
Row-2	7	8	9

Initialing and Accessing a 2D Array: Example-2

Program

```
int data[2][4] = {
{1,2,3,4}, //row 0 with 4 elements
{5,6,7,8}, //row 1 with 4 elements
    };
printf("%d",data[0][0]); //1
printf("%d",data[0][1]); //2
printf("%d",data[0][2]); //3
printf("%d\n",data[0][3]); //4

printf("%d",data[1][0]); //5
printf("%d",data[1][1]); //6
printf("%d",data[1][2]); //7
printf("%d",data[1][3]); //8
```

```
// data[2][4] can be initialized like this also
int data[2][4]={{1,2,3,4},{5,6,7,8}};
```

	Col-0	Col-1	Col-2	Col-3
Row-0	1	2	3	4
Row-1	5	6	7	8

Read(Scan) 2D Array Elements

Program

```
void main(){
    int data[3][3],i,j;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Enter array element=");
            scanf("%d",&data[i][j]);
        }
    }
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d",data[i][j]);
        }
        printf("\n");
    }
}
```

	Column-0	Column-1	Column-2
Row-0	1	2	3
Row-1	4	5	6
Row-2	7	8	9

Output

```
Enter array element=1
Enter array element=2
Enter array element=3
Enter array element=4
Enter array element=5
Enter array element=6
Enter array element=7
Enter array element=8
Enter array element=9
123
456
789
```

Develop a program to count number of positive, negative and zero elements from 3 X 3 matrix

Program

```
void main(){
    int data[3][3],i,j,pos=0,neg=0,zero=0;
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("Enter array element=");
            scanf("%d",&data[i][j]);
            if(data[i][j]>0)
                pos=pos+1;
            else if(data[i][j]<0)
                neg=neg+1;
            else
                zero=zero+1;
        }
    }
    printf("positive=%d,negative=%d,zero=%d",pos,neg,zero);
}
```

Output

```
Enter array element=9
Enter array element=5
Enter array element=6
Enter array element=-3
Enter array element=-7
Enter array element=0
Enter array element=11
Enter array element=13
Enter array element=8
positive=6,negative=2,zero=1
```

Practice Programs

1. Develop a program to perform addition of two matrix.
2. Develop a program to perform multiplication of two matrix.

String (Character Array)

Definition: String

- A String is a one-dimensional array of characters terminated by a `null('\0')`.

```
char name[10];
```

[0]	[1]	[2]	…	[9]

- Each character in the array occupies one byte of memory, and the last character must always be `null('\0')`.
- The termination character `('\0')` is important in a string to identify where the string ends.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
name[10]	D	A	R	S	H	A	K	\0		

Declaring & Initializing String

Declaration

```c
char name[10];
```

Initialization method 1:

```c
char name[10]={'D','A','R','S','H','A',‘K','\0'};
```

Initialization method 2:

```c
char name[10]="DARSHAK";
//'\0' will be automatically inserted at the end in this type of declaration.
```

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
name[10]	D	A	R	S	H	A	k	\0		

Read String: scanf()

Program

```c
void main()
{
    char name[10];
    printf("Enter name:");
    scanf("%s",name);
    printf("Name=%s",name);
}
```

Output

```
Enter name: Darshak
Name=Darshak
```

Output

```
Enter name: CE Darshak
Name=CE
```

- There is no need to use address of `(&)` operator in `scanf` to store a string.
- As string `name` is an array of characters and the name of the array, i.e., `name` indicates the base address of the string (character array).
- `scanf()` terminates its input on the first whitespace(space, tab, newline etc.) encountered.

Read String: gets()

Program

```c
#include<stdio.h>
void main()
{
    char name[10];
    printf("Enter name:");
    gets(name); //read string including white spaces
    printf("Name=%s",name);
}
```

Output

```
Enter name:ITM Institute
Name=ITM Institute
```

- `gets()`: Reads characters from the standard input and stores them as a string.
- `puts()`: Prints characters from the standard.
- `scanf()`: Reads input until it encounters whitespace, newline or End Of File`(EOF)` whereas `gets()` reads input until it encounters newline or End Of File`(EOF)`.
- `gets()`: Does not stop reading input when it encounters whitespace instead it takes whitespace as a string.

String Handling Functions : strlen()

- C has several inbuilt functions to operate on string. These functions are known as string handling functions.
- strlen(s1): returns length of a string in integer

Program

```c
#include <stdio.h>
#include <string.h> //header file for string functions
void main()
{
    char s1[10];
    printf("Enter string:");
    gets(s1);
    printf("%d",strlen(s1)); // returns length of s1 in integer
}
```

Output

```
Enter string: CE Darshak
10
```

String Handling Functions: strcmp()

- `strcmp(s1,s2):` Returns `0` if `s1` and `s2` are the same.
- Returns less than `0` if `s1<s2.`
- Returns greater than `0` if `s1>s2`.

Program

```c
void main()
{
    char s1[10],s2[10];
    printf("Enter string-1:");
    gets(s1);
    printf("Enter string-2:");
    gets(s2);
    if(strcmp(s1,s2)==0)
       printf("Strings are same");
    else
       printf("Strings are not same");
}
```

Output

```
Enter string-1:Computer
Enter string-2:Computer
Strings are same
```

Output

```
Enter string-1:Computer
Enter string-2:Computer
Strings are same
```

String Handling Functions

For examples consider: `char s1[]="Their",s2[]="There";`

Syntax	Description
`strcpy(s1,s2)`	Copies 2nd string to 1st string. `strcpy(s1,s2)` copies the string `s2` in to string `s1` so `s1` is now “There”. `s2` remains unchanged.
`strcat(s1,s2)`	Appends 2nd string at the end of 1st string. `strcat(s1,s2);` a copy of string `s2` is appended at the end of string `s1`. Now `s1` becomes “TheirThere”
`strchr(s1,c)`	Returns a pointer to the first occurrence of a given character in the string `s1`. `printf("%s",strchr(s1,'i'));` Output : ir
`strstr(s1,s2)`	Returns a pointer to the first occurrence of a given string `s2` in string `s1`. `printf("%s",strstr(s1,"he"));` Output : heir

String Handling Functions (Cont…)

For examples consider: char s1[]="Their",s2[]="There";

Syntax	Description
strrev(s1)	Reverses given string. strrev(s1); makes string s1 to “riehT”
strlwr(s1)	Converts string s1 to lower case. printf("%s",strlwr(s1)); Output : their
strupr(s1)	Converts string s1 to upper case. printf("%s",strupr(s1)); Output : THEIR

strncpy(s1,s2,n)	Copies first n character of string s2 to string s1
	s1=""; s2="There";
	strncpy(s1,s2,2);
	printf("%s",s1); Output : Th

strncat(s1,s2,n)	Appends first n character of string s2 at the end of string s1. strncat(s1,s2,2); printf("%s", s1); Output : TheirTh

String Handling Functions (Cont…)

For examples consider: `char s1[]="Their",s2[]="There";`

Syntax	Description
`strncmp(s1,s2,n)`	Compares first n character of string `s1` and `s2` and returns similar result as `strcmp()` function. `printf("%d",strcmp(s1,s2,3));` Output : 0
`strrchr(s1,c)`	Returns the last occurrence of a given character in a string `s1`. `printf("%s",strrchr(s2,'e'));` Output : ere

Pointer

{C}
Programming

What is Pointer?

- A normal variable is used to store value.
- A pointer is a variable that store address / reference of another variable.
- Pointer is derived data type in C language.
- A pointer contains the memory address of that variable as their value. Pointers are also called address variables because they contain the addresses of other variables.

Declaration & Initialization of Pointer

Syntax

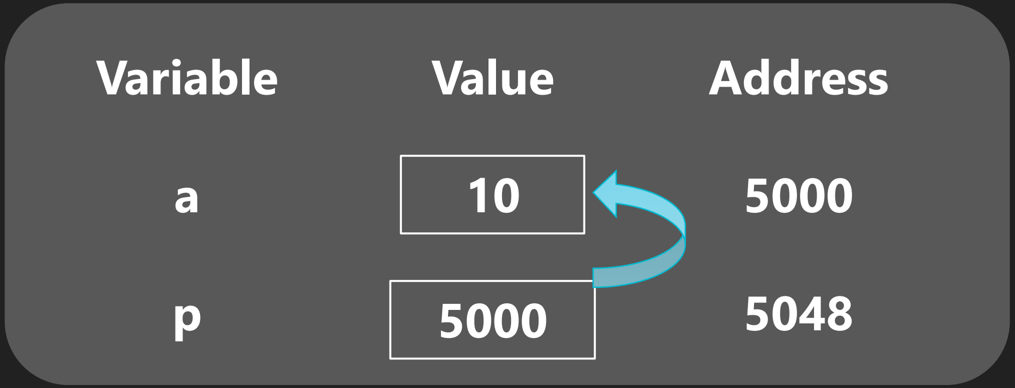
```
datatype *ptr_variablename;
```

Example

```
void main()
{
    int a=10, *p; // assign memory address of a
    to pointer variable p
    p = &a;
    printf("%d %d %d", a, *p, p);
}
```

Output

```
10 10 5000
```



Variable	Value	Address
a	10	5000
p	5000	5048

- p is integer pointer variable
- & is address of or referencing operator which returns memory address of variable.
- * is indirection or dereferencing operator which returns value stored at that memory address.
- & operator is the inverse of * operator
- x = a is same as x = *(&a)

Why use Pointer?

- C uses pointers to create dynamic data structures, data structures built up from blocks of memory allocated from the heap at run-time. Example linked list, tree, etc.
- C uses pointers to handle variable parameters passed to functions.
- Pointers in C provide an alternative way to access information stored in arrays.
- Pointer use in system level programming where memory addresses are useful. For example shared memory used by multiple threads.
- Pointers are used for file handling.
- This is the reason why C is versatile.

Pointer to Pointer – Double Pointer

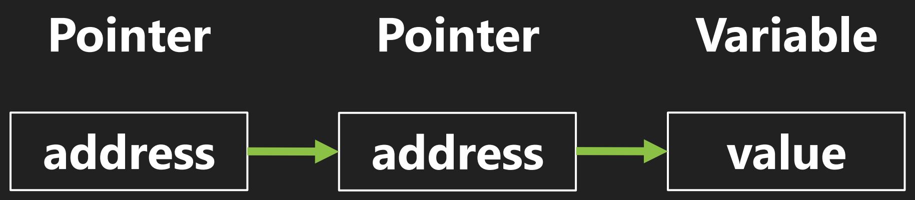
- Pointer holds the address of another variable of same type.
- When a pointer holds the address of another pointer then such type of pointer is known as pointer-to-pointer or double pointer.
- The first pointer contains the address of the second pointer, which points to the location that contains the actual value.

Syntax

```
datatype **ptr_variablename;
```

Example

```
int **ptr;
```



Write a program to print variable, address of pointer variable and pointer to pointer variable.

W o print Odd numbers between 1 to n

Program

```c
#include <stdio.h>
int main () {
    int var;
    int *ptr;
    int **pptr;
    var = 3000;
    ptr = &var; // address of var
    pptr = &ptr; // address of ptr using address of operator &
    printf("Value of var = %d\n", var );
    printf("Value available at *ptr = %d\n", *ptr );
        printf("Value available at **pptr = %d\n", **pptr);
        return 0;
}
```

Output

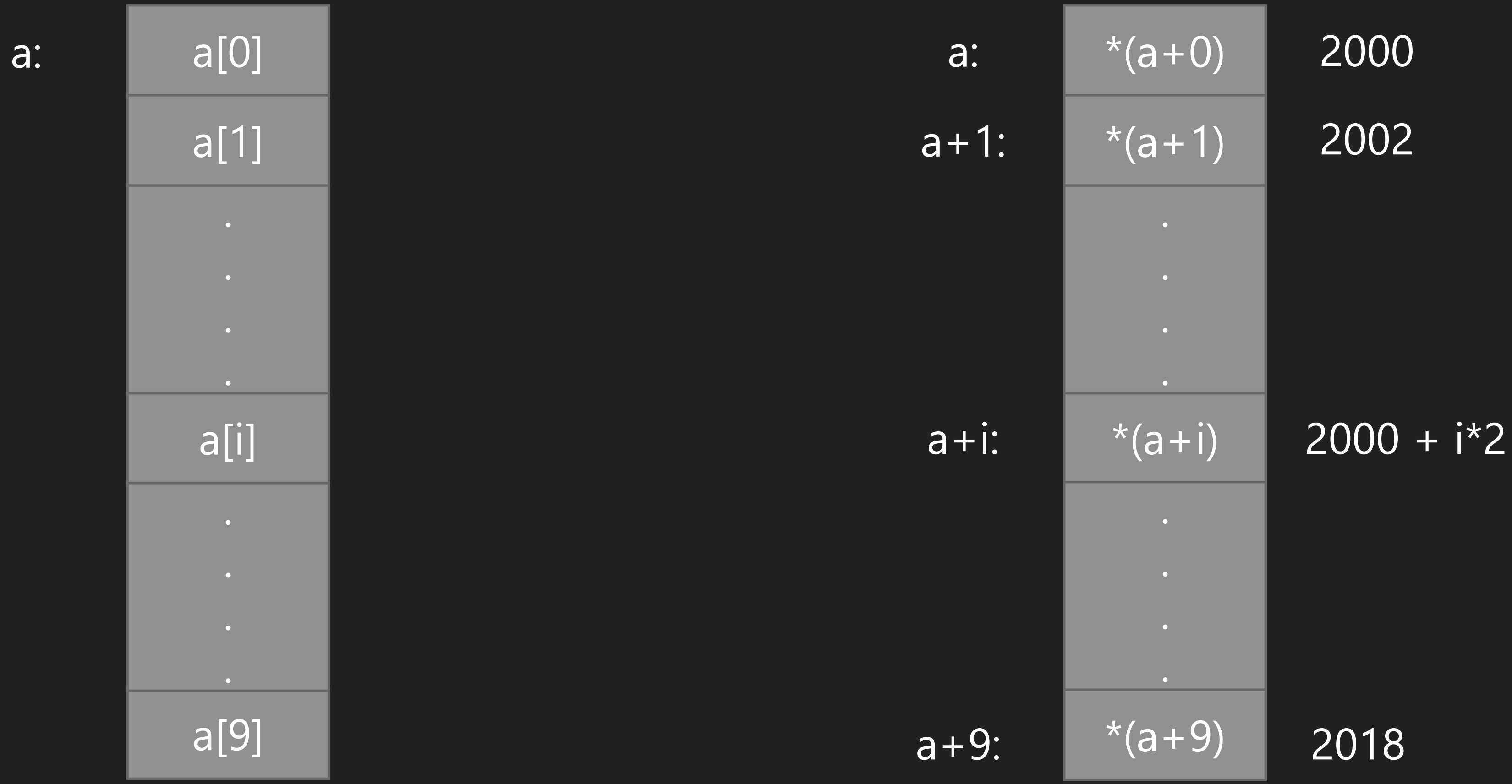
```
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```

Relation between Array & Pointer

- When we declare an array, compiler allocates continuous blocks of memory so that all the elements of an array can be stored in that memory.
- The address of first allocated byte or the address of first element is assigned to an array name.
- Thus array name works as pointer variable.
- The address of first element is also known as base address.

Relation between Array & Pointer – Cont.

- Example: `int a[10], *p;`
- a[0] is same as *(a+0), a[2] is same as *(a+2) and a[i] is same as *(a+i)



a:	a[0]
	a[1]
	. . . .
	a[i]
	. . . .
	a[9]

a:	*(a+0)	2000
a+1:	*(a+1)	2002
	. . . .	
a+i:	*(a+i)	2000 + i*2
	. . . .	
a+9:	*(a+9)	2018

Array of Pointer

- As we have an array of char, int, float etc, same way we can have an array of pointer.
- Individual elements of an array will store the address values.
- So, an array is a collection of values of similar type. It can also be a collection of references of similar type known by single name.

Syntax

```
datatype *name[size];
```

Example

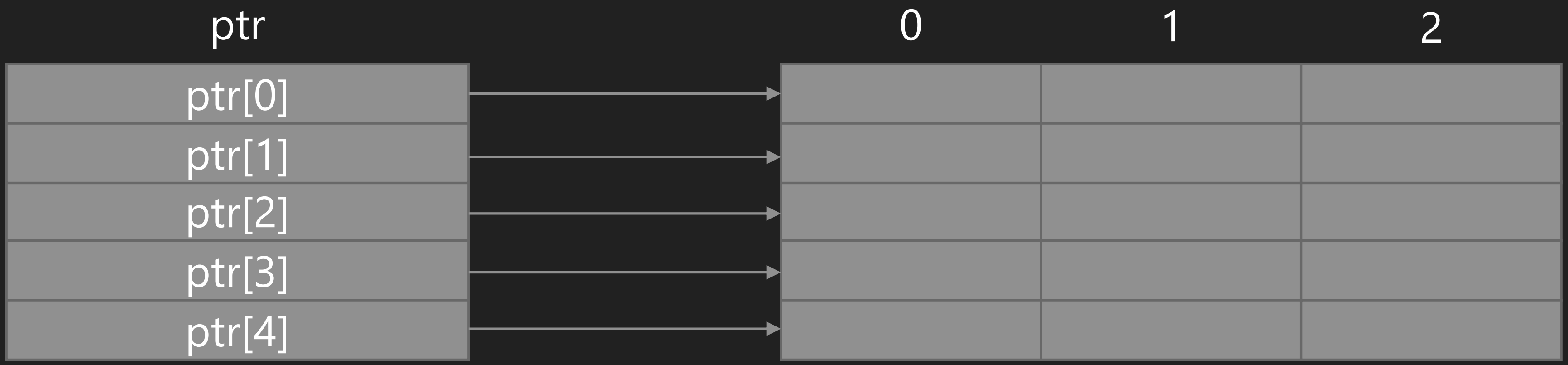
```
int *ptr[5]; //declares an array of integer pointer of size 5
```

Array of Pointer – Cont.

- An array of pointers ptr can be used to point to different rows of matrix as follow:

Example

```
for(i=0; i<5; i++)
{
        ptr[i]=&mat[i][0];
}
```



- By dynamic memory allocation, we do not require to declare two-dimensional array, it can be created dynamically using array of pointers.

Write a program to swap value of two variables using pointer / call by reference.

W e print Odd numbers between 1 to n

Program

```c
int main()
{
    int num1,num2;
    printf("Enter value of num1 and num2: ");
    scanf("%d %d",&num1, &num2);

//displaying numbers before swapping
    printf("Before Swapping: num1 is: %d, num2 is: %d\n",num1,num2);

//calling the user defined function swap()
    swap(&num1,&num2);

//displaying numbers after swapping
    printf("After Swapping: num1 is: %d, num2 is: %d\n",num1,num2);
    return 0;
}
```

Output

```
Enter value of num1 and num2: 5
10
Before Swapping: num1 is: 5, num2 is: 10
After  Swapping: num1 is: 10, num2 is: 5
```

Pointer and Function

- Like normal variable, pointer variable can be passed as function argument and function can return pointer as well.
- There are two approaches to passing argument to a function:
  - Call by value
  - Call by reference / address

Call by Value

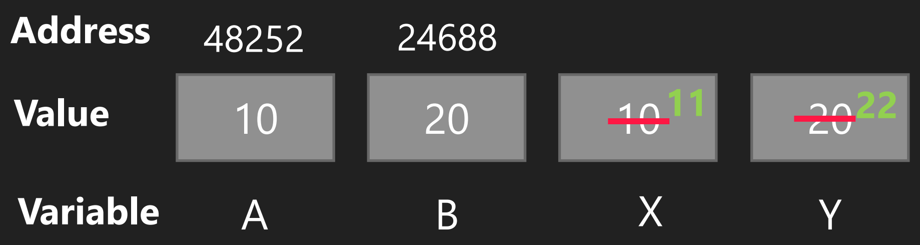
- In this approach, the values are passed as function argument to the definition of function.

Program

```c
#include<stdio.h>
void fun(int,int);
int main()
{
    int A=10,B=20;
    printf("\nValues before calling %d, %d",A,B);
    fun(A,B);
    printf("\nValues after calling %d, %d",A,B);
    return 0;
}
void fun(int X,int Y)
{
    X=11;
    Y=22;
}
```

Output

```
Values before calling 10, 20
Values after calling 10, 20
```



Address	48252	24688		
Value	10	20	~~10~~11	~~20~~22
Variable	A	B	X	Y

Call by Reference / Address

- In this approach, the references / addresses are passed as function argument to the definition of function.

Program

```
#include<stdio.h>
void fun(int*,int*);
int main()
{
    int A=10,B=20;
    printf("\nValues before calling %d, %d",A,B);
    fun(&A,&B);
    printf("\nValues after calling %d, %d",A,B);
    return 0;
}
void fun(int *X,int *Y)
{
    *X=11;
    *Y=22;
}
```

Output

```
Values before calling 10, 20
Values after  calling 11, 22
```

Address	48252	24688		
Value	~~10~~ 11	~~20~~ 22	48252	24688
Variable	A	B	*X	*Y

Pointer to Function

- Every function has reference or address, and if we know the reference or address of function, we can access the function using its reference or address.
- This is the way of accessing function using pointer.

Syntax

```
return-type (*ptr-function)(argument list);
```

- return-type: Type of value function will return.
- argument list: Represents the type and number of value function will take, values are sent by the calling statement.
- (*ptr-function): The parentheses around *ptr-function tells the compiler that it is pointer to function.
- If we write *ptr-function without parentheses then it tells the compiler that ptr-function is a function that will return a pointer.

Write a program to sum of two numbers using pointer to function.

W o print Odd numbers between 1 to n

Program

```c
#include<stdio.h>
int Sum(int,int);
int (*ptr)(int,int);
int main()
{
    int a,b,rt;
    printf("\nEnter 1st number : ");
    scanf("%d",&a);
    printf("\nEnter 2nd number : ");
    scanf("%d",&b);
    ptr = Sum;
    rt = (*ptr)(a,b);
    printf("\nThe sum is : %d",rt);
    return 0;
}
int Sum(int x,int y)
{
        return x + y;
}
```

Output

```
Enter 1st number : 5

Enter 2nd number : 10

The sum is : 15
```

Practice Programs

1. Write a C program to print the address of variable using pointer.
2. Write a C a program to swap two elements using pointer.
3. Write a C a program to print value and address of a variable
4. Write a C a program to calculate sum of two numbers using pointer
5. Write a C a program to swap value of two numbers using pointer
6. Write a C a program to calculate sum of elements of an array using pointer
7. Write a C a program to swap value of two variables using function
8. Write a C a program to print the address of character and the character of string using pointer
9. Write a C a program for sorting using pointer

Thank you