NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENC

CL 1004 - Object Oriented Programming Lab
Semester: Spring 2022
Lab 01

Instructor: Muhammad Sudais
email: sudaismsm@gmail.com

Outline:

- Introduction To Object Oriented Programming
- Difference between C vs C++
- Introduction Of IDE
- Skeleton Of C++ Program,
- Headers File in- C and C++
- Use of Namespace with standard library
- Basic I/O In C++
- Debugging
- Array
- Pointer
- Pointers to array
- Double Pointers
- Dynamic Memory Management (new and delete operators)
- Dynamic Memory Management

Introduction To Object Oriented Programming:

- Object oriented programming is a way of solving complex problems by breaking them into smaller problems using objects.
- Before Object Oriented Programming (commonly referred as OOP), programs were written in procedural language, they were nothing but a long list of instructions. On the other hand, the OOP is all about creating objects that can interact with each other, this makes it easier to develop programs in OOP

So what is an object oriented program exactly?

It is a program that contains objects, of course, which have certain properties and have methods linked to them. These methods are used to obtain a programming result.

WHY IS OOP NEEDED?

Problems with Procedural Languages

– Functions have unrestricted access to global data

Unrelated Functions and data.

Before Object Oriented Programming programs were viewed as procedures that accepted data and produced an output.

DIFFERENCE BETWEEN C AND C++

The key differences include:

C	C++
It is a structural or procedural programming language.	It is an object oriented programming language.
Emphasis is on procedure or steps to solve a problem.	Emphasis is on objects rather than procedure
Functions are the fundamental building blocks.	Objects are the fundamental building blocks.
In C, the data is not secured.	Data is hidden and can't be accessed by external functions.
C uses scanf() and printf() functions for standard input a output.	C uses cin>> and cout<< functions for standard inp output.
In C, namespace feature is absent.	In C++, namespace feature is present.
C program file is saved with .C extension.	C++ program file is saved with .CPP extension.

The main difference between both these languages is C is a procedural programming language and does not support classes and objects, while C++ is a combination of both procedural and object-oriented programming languages

About IDE

We can use Dev C++ or VS code or code block.

EXPLANATION OF BASIC C++ PROGRAM

/* Comments can also be written starting with a slash followed by a star, and ending with a star followed by a slash. As you can see, comments written in this way can span more than one line. */ /* Programs should ALWAYS include plenty of comments! */ /* This program prints the table of entered number */

Headers File in- C and C++

In C++, all the header files may or may not end with the .h extension but in C, all the header files must necessarily begin with the.h extension.

- A header file in C/C++ contains:
- Function definitions
- Data type definitions

1. Standard library header files: These are the pre-existing header files already available in the C/C++ compiler.
2. User-defined header files: Header files starting #define can be designed by the user.

#include<string.h>
#include<stdio.h>
#include<iostream>
#include<factorial.h>
	The #include is a preprocessor command that tells t compiler to include the content.

Using Namespace std;

- Namespaces allow to group entities like classes, objects and functions under a name. This way the global scope can be divided into **"sub-scopes"**, each one with its own name.
- The names cout and endl belong to the std namespace. They can be referenced via fully qualified names **std::cout** and **std::endl**, or simply as cout and endl with a **"using namespace std;"** statement.

Basic I/O Function

- Every **C program** should necessarily contain the header file **<stdio.h>** which stands for standard input and output used to take input with the help of **scanf()** function and display the output using **printf()** function.
- **C++ program** should necessarily contain the header file **<iostream>** which stands for input and output stream used to take input with the help of **"cin>>"** function and display the output using **"cout<<"** function.

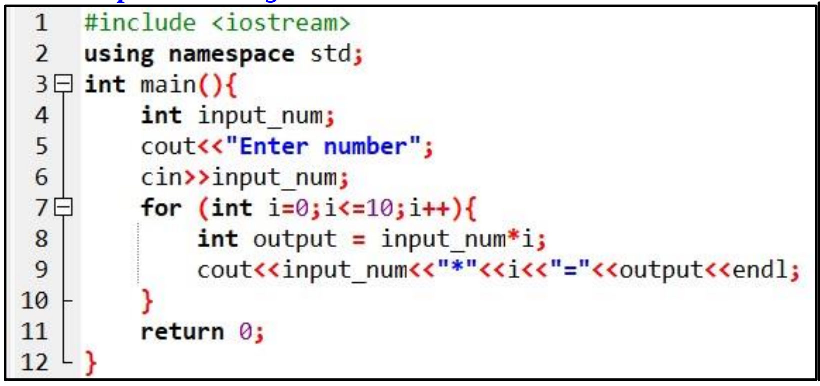
Output using cout
Cout is an object
Corresponds to standard output stream
<< is called insertion or input operator

Input With cin
Cin is an object
Corresponds to standard input stream
>> is called extraction or get from operator

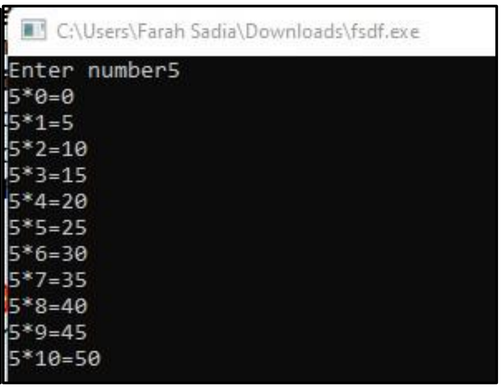
return 0;

The return value of 0 indicates normal termination; while non-zero (typically 1) indicates abnormal termination. C++ compiler will automatically insert a "return 0;" at the end of the the main () function, thus, it statement can be omitted.

An Example C++ Program



```
#include <iostream>
using namespace std;
int main(){
    int input_num;
    cout<<"Enter number";
    cin>>input_num;
    for (int i=0;i<=10;i++){
        int output = input_num*i;
        cout<<input_num<<"*"<<i<<"="<<output<<endl;
    }
    return 0;
}
```



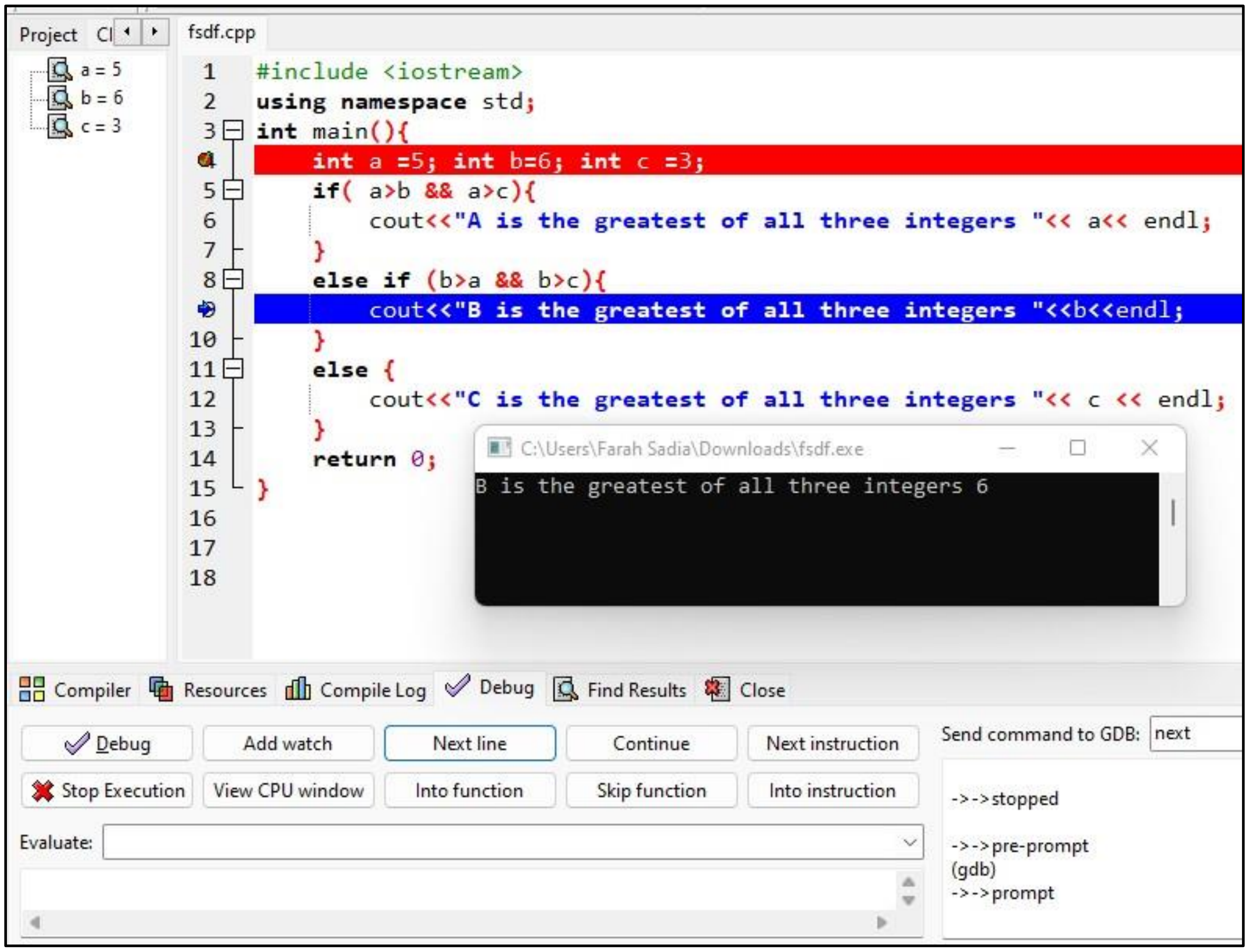
```
Enter number5
5*0=0
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
5*10=50
```

Debugging in C++:

Steps to Remember:

Consider the following Code as an example

The following coding example illustrates how does a debugger executes the code, steps into which method's body, how it steps out and when does it step over a function.



```
#include <iostream>
using namespace std;
int main(){
    int a =5; int b=6; int c =3;
    if( a>b && a>c){
        cout<<"A is the greatest of all three integers "<< a<< endl;
    }
    else if (b>a && b>c){
        cout<<"B is the greatest of all three integers "<<b<<endl;
    }
    else {
        cout<<"C is the greatest of all three integers "<< c << endl;
    }
    return 0;
}
```

```
B is the greatest of all three integers 6
```

a = 5
b = 6
c = 3

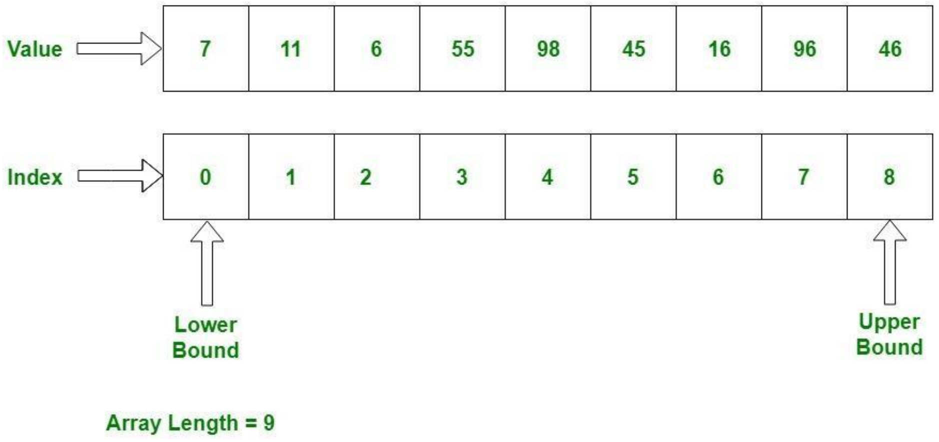
Send command to GDB: next

```
->->stopped

->->pre-prompt
(gdb)
->->prompt
```

Array

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.



Types

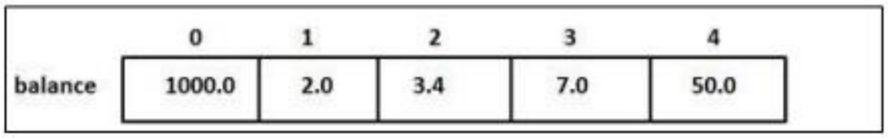
- One-dimensional array – Vectors
- Two-dimensional array – Matrix

Declaring Arrays

type arrayName [arraySize];

double balance[10]; **Initializing Arrays**

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

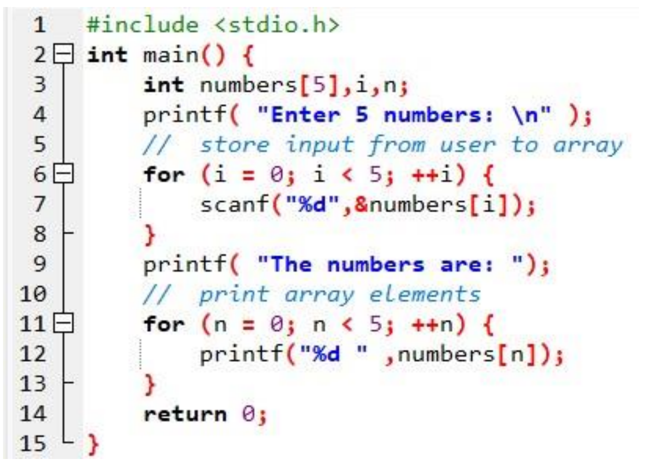


Accessing Array- C vs C++ double

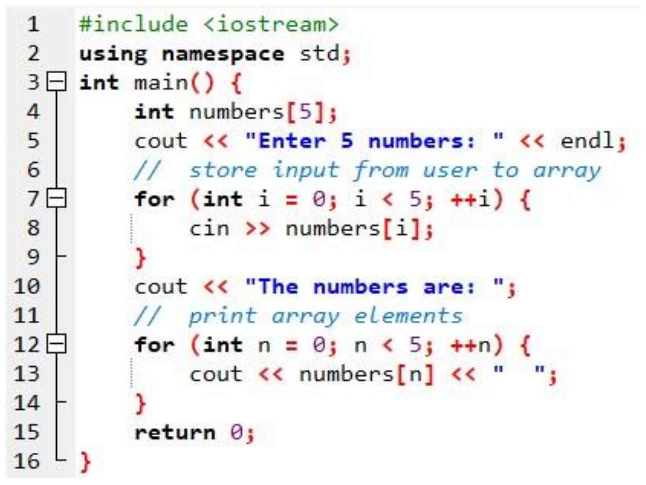
salary = balance [3]

Output = 7.0

An Example C/C++ Program



```c
#include <stdio.h>
int main() {
    int numbers[5],i,n;
    printf( "Enter 5 numbers: \n" );
    //  store input from user to array
    for (i = 0; i < 5; ++i) {
        scanf("%d",&numbers[i]);
    }
    printf( "The numbers are: ");
    //  print array elements
    for (n = 0; n < 5; ++n) {
        printf("%d " ,numbers[n]);
    }
    return 0;
}
```

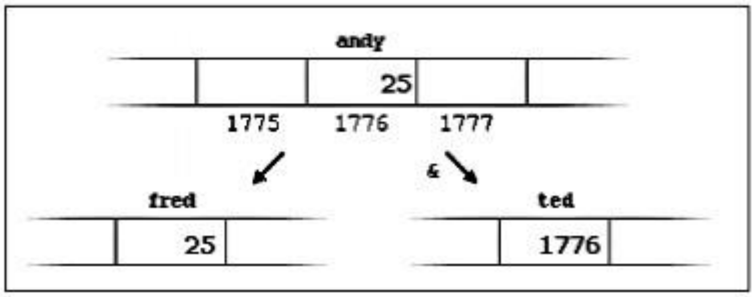


```cpp
#include <iostream>
using namespace std;
int main() {
    int numbers[5];
    cout << "Enter 5 numbers: " << endl;
    //  store input from user to array
    for (int i = 0; i < 5; ++i) {
        cin >> numbers[i];
    }
    cout << "The numbers are: ";
    //  print array elements
    for (int n = 0; n < 5; ++n) {
        cout << numbers[n] << "  ";
    }
    return 0;
}
```

Pointer

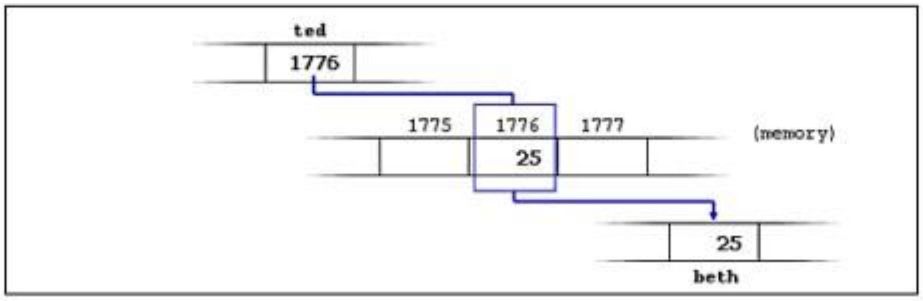
- Pointer is a variable whose value is a memory address. Normally, a variable directly contains a specific value.
- A pointer contains the memory address of a variable that, in turn, contains a specific value.
- In this sense, a variable name directly references a value, and a pointer indirectly references a value. ● The address that locates a variable within memory is what we call a reference to that variable. **ted = &andy;**
- Assume that andy is placed during runtime in the memory address 1776. **andy = 25; fred = andy; ted = &andy;**

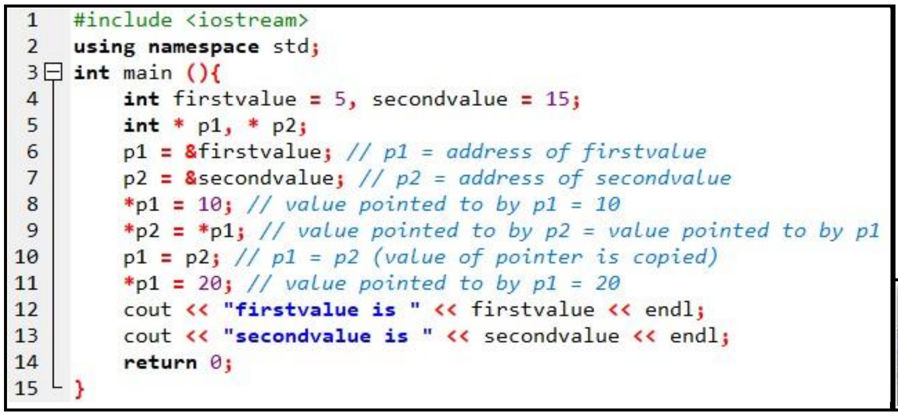
The values contained in each variable after the execution of this, are shown in the following diagram:



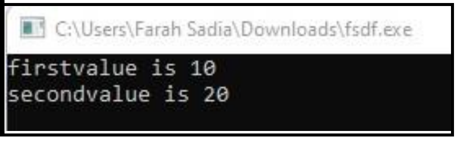
Dereference Operator (*)

- The asterisk (*) acts as a dereference operator and that can be translated to "value pointed by". **beth = *ted;**
- We could read as: "beth equal to value pointed by ted", beth would take the value 25, since ted is 1776, and the value pointed by 1776 is 25.





```
#include <iostream>
using namespace std;
int main (){
    int firstvalue = 5, secondvalue = 15;
    int * p1, * p2;
    p1 = &firstvalue; // p1 = address of firstvalue
    p2 = &secondvalue; // p2 = address of secondvalue
    *p1 = 10; // value pointed to by p1 = 10
    *p2 = *p1; // value pointed to by p2 = value pointed to by p1
    p1 = p2; // p1 = p2 (value of pointer is copied)
    *p1 = 20; // value pointed to by p1 = 20
    cout << "firstvalue is " << firstvalue << endl;
    cout << "secondvalue is " << secondvalue << endl;
    return 0;
}
```



C:\Users\Farah Sadia\Downloads\fsdf.exe

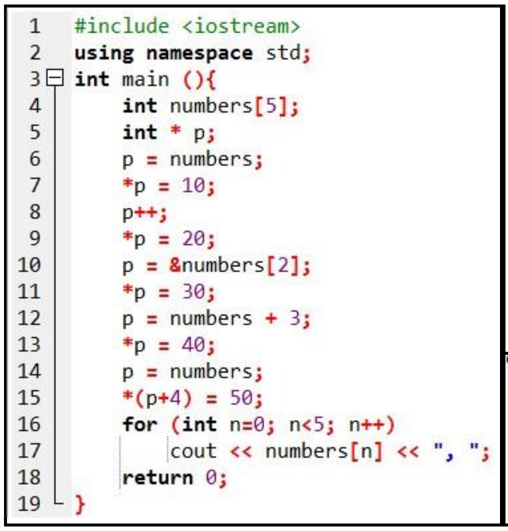
```
firstvalue is 10
secondvalue is 20
```

Program as example

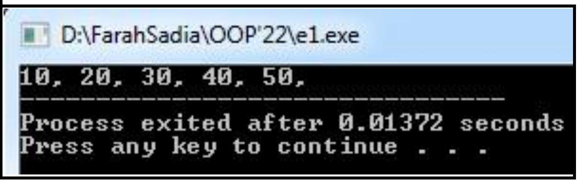
Pointers and Arrays

- The identifier of an array is equivalent to the address of its first element, as a pointer is equivalent to the address of the first element that it points to.
- EXAMPLE: Assume the following declaration: **int numbers [20];**
  int * p;
- The following assignment operation would be valid: **p = numbers;**

Program as example



```
#include <iostream>
using namespace std;
int main (){
    int numbers[5];
    int * p;
    p = numbers;
    *p = 10;
    p++;
    *p = 20;
    p = &numbers[2];
    *p = 30;
    p = numbers + 3;
    *p = 40;
    p = numbers;
    *(p+4) = 50;
    for (int n=0; n<5; n++)
        cout << numbers[n] << ", ";
    return 0;
}
```



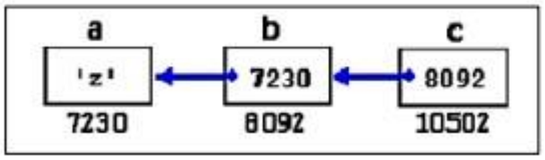
```
D:\FarahSadia\OOP'22\e1.exe
10, 20, 30, 40, 50,
--------------------------------
Process exited after 0.01372 seconds
Press any key to continue . . .
```

Pointers to Pointers (Double Pointer)

- C++ allows the use of pointers that point to pointers, that in turn, point to data (or even to other pointers). In order to do that, we only need to add an asterisk (*) for each level of reference in their declarations:

  char a;
  char * b;
  char **
  c; a = 'z';
  b = &a;
  c = &b;

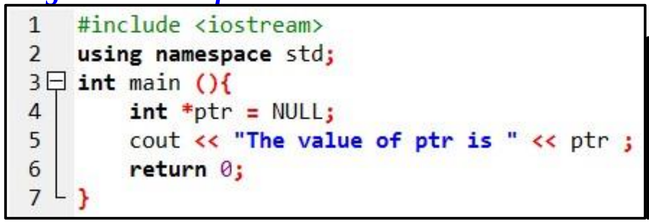
This, supposing the randomly chosen memory locations for each variable of 7230, 8092 and 10502, could be represented as:



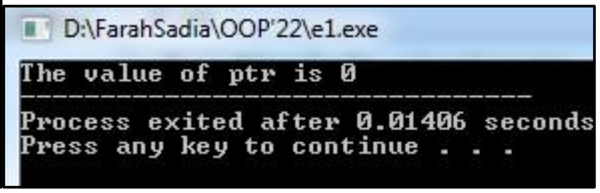
Null Pointer

- A null pointer is a regular pointer of any pointer type which has a special value that indicates that it is not pointing to any valid reference or memory address.

  Program as example



```
#include <iostream>
using namespace std;
int main (){
    int *ptr = NULL;
    cout << "The value of ptr is " << ptr ;
    return 0;
}
```



```
D:\FarahSadia\OOP'22\e1.exe
The value of ptr is 0
--------------------------------
Process exited after 0.01406 seconds
Press any key to continue . . .
```

Memory Allocation

- There are two ways through which memory can be allocated to a variable. They are static allocation of memory and dynamic memory allocation. So far, we have declared variables and arrays statically. This means at compile time memory is allocated to a variable or array.

  Example

  - static memory allocation int a = 10;
  - Dynamic Memory Allocation: if we want a variable amount of memory that can only be determined during runtime.
  - When we talk about dynamic memory allocation or runtime memory allocation, two keywords are important in C++. They are new and deleted.

- ○ new and new[ ] operator new operator is used to dynamically allocate memory. When we allocate memory using a new keyword a pointer is needed. This is so because new allocates memory dynamically and it returns address to allocated memory and this returned memory address is stored in a pointer variable.

General Syntax for dynamically memory allocation for a variable

Example: **int *ptr;**

ptr = new int; //dynamically memory allocated for an integer and address of allocated memory is stored

Example: **int *ptr; ptr = new int;**

***ptr = 10;** // dereferencing pointer

General Syntax for dynamically memory allocation for an array

Example:

Pointer = new type[size];

Example **int *ptr;**

ptr = new int[10]; // dynamically memory allocated for an integer array of size 10 and address of first element of array is stored in ptr

Accessing value stored at the address in pointer variable (array). This can be done in two ways.

1. Use pointer subscript notation.

i.e ptr[0] = 10, ptr[1] = 20 etc.

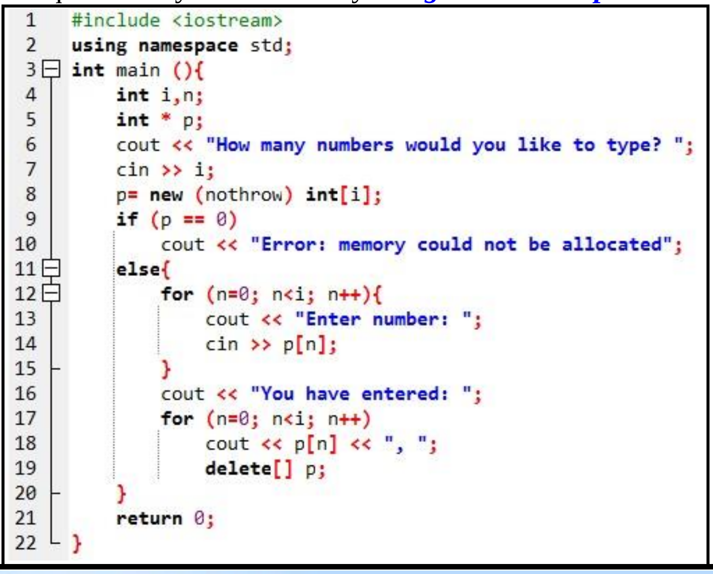
OR

2. Use a dereferencing operator.

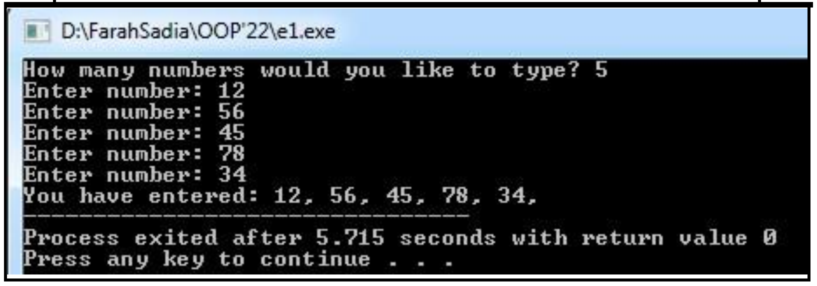
i.e *(ptr+0) = 10, *(ptr+1) = 20; Delete

And Delete[ ] Operator

- To free dynamic memory after it is used, delete operator is used so that the memory becomes available again for other requests of dynamic memory. ***Program as example***



```
#include <iostream>
using namespace std;
int main (){
    int i,n;
    int * p;
    cout << "How many numbers would you like to type? ";
    cin >> i;
    p= new (nothrow) int[i];
    if (p == 0)
        cout << "Error: memory could not be allocated";
    else{
        for (n=0; n<i; n++){
            cout << "Enter number: ";
            cin >> p[n];
        }
        cout << "You have entered: ";
        for (n=0; n<i; n++)
            cout << p[n] << ", ";
            delete[] p;
    }
    return 0;
}
```



D:\FarahSadia\OOP'22\e1.exe

```
How many numbers would you like to type? 5
Enter number: 12
Enter number: 56
Enter number: 45
Enter number: 78
Enter number: 34
You have entered: 12, 56, 45, 78, 34,
--------------------------------
Process exited after 5.715 seconds with return value 0
Press any key to continue . . .
```

Exercise

Question # 01:
Write a program for generating the output given below against each input. You can use any of these operator for each input : +=, -= and *=.

Input:
10
50
2
30
34

Output:
100
40
4
90
24

Question # 02:
Write a program to perform the following 2x2 matrix operation.

- **Mat A + 2(Mat B) – Mat C**

Input:

Mat A:

2	3
3	2

Mat B:

2	3
4	5

Mat C:

1	0
3	1

Hint:
2(Mat B):

4	6
8	10

You can use arrays for storing and displaying values of Matrices.

Output:

5	9
8	11

Question # 03:
Write a function named "array_mix " that takes two array of same length as argument and interchanges the values of arrays if the index is odd number that are stored in those arguments. The function should return no value but it must print the values of output array on each iteration.

Array 1:

2
3
6
7

Array 2:

1
4
5
8

Output:
Array1: 2, 4, 6, 8
Array2: 1, 3, 5, 7

Question # 04:
Create the equivalent of a multi-function calculator. The program should ask the user to enter a number, an operator, and another number. (Use floating point.) It should then carry out the specified arithmetical operation. Use a switch statement to select the operation and display the result.
Your have to use C++ functions inside each switch case to perform the operations.
Following are the functions:

1. add(int a, int b)
2. subtract(int a, int b)
3. multiply(int a, int b)
4. divide(int a, int b)
5. remainder(int a, int b)
6. square(int a)
7. cube(int a)
8. roundOff(int a)

Question # 05:
Write a function name factorial(int a) to calculate the factorial of a number.
Input: 4

Calculation: 4x3x2x1

Ouput: 24

Question # 07:
Write a program that displays the following menu for the food items available to take order from the customer:

B= Burger (Rs. 200)
F= French Fries (Rs. 50)
P= Pizza (Rs. 500)
S= Soft Drink (Rs. 50)

- The customer can order any combination of available food. The program first asks the user if he wants a burger
  - If yes ask him the quantity.
  - Else burger quantity is zero

Same goes for other items.

- The program will calculate the total charges, considering the price and quantity.
- The program should display the charges without the tax and discount for the order.

- The program will then ask for the discount code from user. If the input (code) is “DISCOUNT20”. Give him 20% off(remove 20% amount from charges).
- Add 10% tax on final amount.
- Display the final charges.