Lecture 19

DMA & Filing

INFORMATION TECHNOLOGY UNIVERSITY

QUIZ

سورة طه پارہ: 16

قَالَ رَبِّ اشْرَحْ لِيْ صَدْرِيْ ۙ ﴿٢٥﴾

[قَالَ: اس نے کہا] [رَبِّ: اے رب !] [اشْرَحْ: کھول دے] [لِيْ: میرے لیے] [صَدْرِيْ : میرا سینہ]

وَيَسِّرْ لِيْٓ اَمْرِيْ ۙ ﴿٢٦﴾

[وَيَسِّرْ: اور آسان کر دے] [لِيْ: میرے لیے] [اَمْرِيْ : میرا کام]

وَاحْلُلْ عُقْدَةً مِّنْ لِّسَانِيْ ۙ ﴿٢٧﴾

[وَاحْلُلْ: اور کھول دے] [عُقْدَةً: گرہ] [مِّنْ: سے] [لِّسَانِيْ: میری زبان]

يَفْقَهُوْا قَوْلِيْ ۖ ﴿٢٨﴾

[يَفْقَهُوْا: وہ سمجھ سکیں] [قَوْلِيْ : میری بات]

QUESTIONS / FEEDBACK / CONCERNS

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC A
SLIDE OF FAME

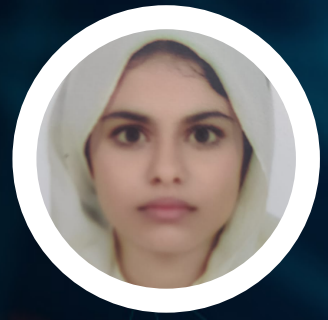
NO ONE
WEEK-1

Muhammad Daniyal Hammad (BSSE23046)
WEEK-2

Syed Hashim Abbas (BSSE23084)
WEEK-3

Umar Ahmad (BSSE23032)
WEEK-4

Umar Ahmad (BSSE23032)
WEEK-5



Fatima Noorulain BSSE23003
WEEK-6

Umar Ahmad (BSSE23032)
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

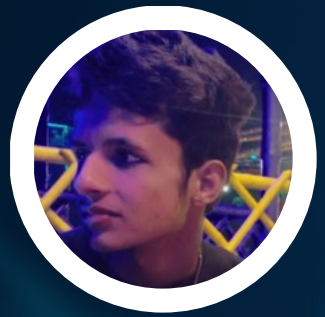
YOUR NAME
WEEK-15

UBS

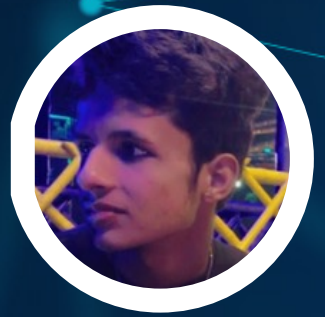
ITU INFORMATION TECHNOLOGY UNIVERSITY

SE SEC B
SLIDE OF FAME

Muhammad Mukarram
BSSE23029
WEEK-1



Muhammad Abdullah
(BSSE23087)
WEEK-2



Muhammad Abdullah
(BSSE23087)
WEEK-3

Fasiha Rohail
(BSSE23041)
WEEK-4

Muhammad Abdullah
(BSSE23087)
WEEK-5

Hazira Azam
BSSE23019
WEEK-6

Jamshaid Ahmed
BSSE23012
WEEK-7

YOUR NAME
WEEK-8

YOUR NAME
WEEK-9

YOUR NAME
WEEK-10

YOUR NAME
WEEK-11

YOUR NAME
WEEK-12

YOUR NAME
WEEK-13

YOUR NAME
WEEK-14

YOUR NAME
WEEK-15

UBS

ITU INFORMATION TECHNOLOGY UNIVERSITY

RECAP

GitHub

Tools (Cygwin, IDE, GitHub)

Approach towards a word problem

Flowcharts

Flowcharts Advantages & Disadvantages

Algorithms

Pseudocode

Numbers Systems (Decimal, Binary, Octal & Hexadecimal)

Ten’s Complement

Twos Complement

main function

Stream in and stream out operators

if else

Functions

Data Types

Arithmetic Operators

Relational Operators

Loops (While, for , do while)

Nested Loops

Switch cases

RECAP

Function Overloading

Scope of variables

Function Prototype and Definition

Default Value in parameters of functions

Parameters by value vs Parameters by Reference

Recursion

Arrays

2D Arrays / Multi Dimensional Arrays

Pointers

ARRAY & POINTERS

```
// arrPtr.cpp
// Outputs addresses and values of array elements.
// -----------------------------------------------

#include <iostream>
using namespace std;

int arr[4] = { 0, 10, 20, 30 };

int main()
{
   cout << "\nAddress and value of array elements:\n"
        << endl;

   for( int i = 0; i < 4; i++ )
      cout << "Address: " << (void*)(arr+i)    // &arr[i]
           << "   Value: " << *(arr+i)         // arr[i]
           << endl;

   return 0;
}
```

POINTERS

```
float v[6] = { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5 },
      *pv, x;

pv = v + 4;
*pv = 1.4;
pv -= 2;
++pv;

x = *pv++;

x += *pv--;

--pv;
```

POINTERS

```
float v[6] = { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5 },
      *pv, x;

pv = v + 4;          // Let pv point to v[4].
*pv = 1.4;           // Assign 1.4 to v[4].
pv -= 2;             // Reset pv to v[2].
++pv;                // Let pv point to v[3].

x = *pv++;           // Assign v[3] to x and
                     // increment pv.
x += *pv--;          // Increment x by v[4] and let
                     // pv point to v[3] again.
--pv;                // Reset pv to v[2].
```

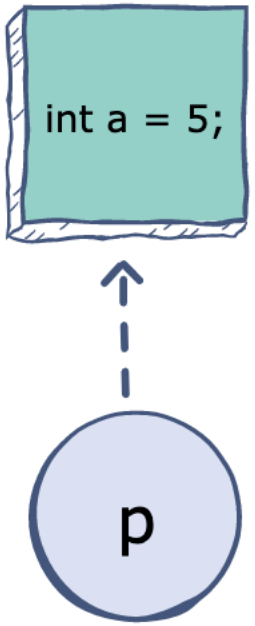
```
// textPtr.cpp
// Using arrays of char and pointers to char
// ---------------------------------------------------
#include <iostream>
using namespace std;

int main()
{
   cout << "Demonstrating arrays of char "
        << "and pointers to char.\n"
        << endl;

   char text[] = "Good morning!",
        name[] = "Bill!";
   char *cPtr = "Hello ";           // Let cPtr point
                                    // to "Hello ".
   cout << cPtr << name << '\n'
        << text << endl;

   cout << "The text \"" << text
        << "\" starts at address " << (void*)text
        << endl;

   cout << text + 6       // What happens now?
        << endl;

   cPtr = name;     // Let cPtr point to name, i.e. *cPtr
                    // is equivalent to name[0]
   cout << "This is the " << *cPtr << " of " << cPtr
        << endl;
   *cPtr = 'k';
   cout << "Bill can not " << cPtr << "!\n" << endl;
   return 0;
}
```

Pointers

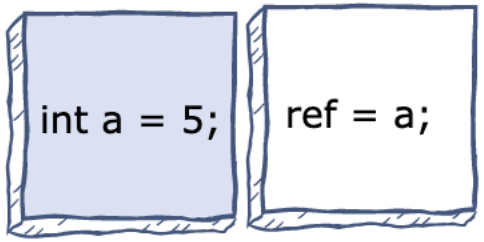
```
#include <iostream>
using namespace std;

void xyz ( int * ptr1, int * ptr2)
{
    int * ptr3 = new int;
    *ptr3 = *ptr1;
    *ptr1 = *ptr2;
    *ptr2 = *ptr3;
}

void printCurrentState ( int * arr, int * arr2)
{
    cout << "******"<<endl;
    cout << " Current arr is   " <<*arr << " and address is " << (void *) arr << endl;
    cout << " Current arr2 is   " <<*arr2 << " and address is " << (void *) arr2 << endl;
}

int main ()
{
    int arr[] = {0,1,2,3,4,5};
    int * arr2 = arr + 2;
    printCurrentState(arr,arr2);
    xyz(arr,arr2);
    printCurrentState(arr,arr2);
    return 0;
}
```

POINTERS VS REFERENCE

- A pointer in C++ is a variable that holds the memory address of another variable.
- A reference is an alias for an already existing variable. Once a reference is initialized to a variable, it cannot be changed to refer to another variable. Hence, a reference is similar to a const pointer.
-



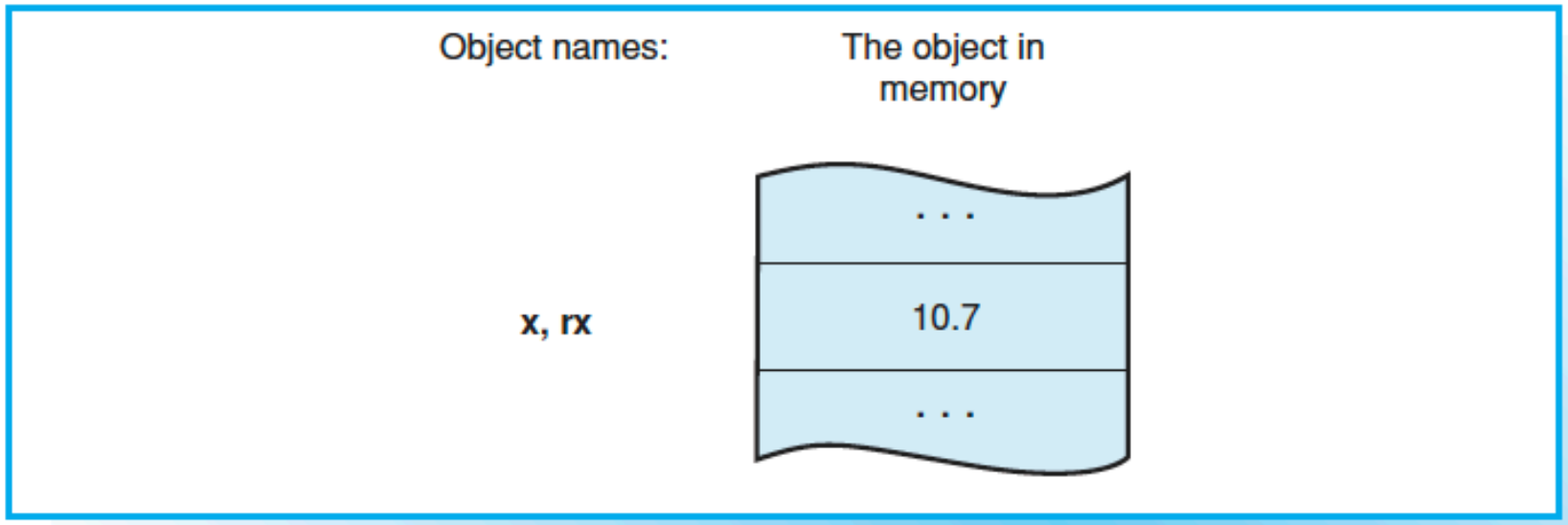
The pointer variable **p** stores the address of the variable **a**; it "points" to the memory location of **a**.



Creating a reference to **a** just makes an alias for it; it does not "point" to **a** by storing its address in a separate memory location

https://www.educative.io/edpresso/differences-between-pointers-and-references-in-cpp

```
float x = 10.7,   &rx = x;
```



REFERENCE

Image Reference : A Complete Guide to Programming in C++

```
// Ref1.cpp
// Demonstrates the definition and use of references.
// ---------------------------------------------------
#include <iostream>
#include <string>
using namespace std;

float x = 10.7F;

int main()
{
   float  &rx = x;
// double &ref = x;

   rx *= 2;

   cout << "   x = " <<  x << endl
        << "  rx = " << rx << endl;
   const float& cref = x;
   cout << "cref = " <<  cref << endl;
// ++cref;
   const string str = "I am a constant string!";
// str = "That doesn't work!";
// string& text = str;
   const string& text = str;
   cout << text << endl;
   return 0;
}
```

REFERENCE

```
// Ref1.cpp
// Demonstrates the definition and use of references.
// ---------------------------------------------------
#include <iostream>
#include <string>
using namespace std;

float x = 10.7F;                                  // Global

int main()
{
   float  &rx = x;              // Local reference to x
// double &ref = x;             // Error: different type!

   rx *= 2;

   cout << "   x = " <<  x << endl     //  x = 21.4
        << "  rx = " << rx << endl;    // rx = 21.4
   const float& cref = x;     // Read-only reference
   cout << "cref = " <<  cref << endl;   // ok!
// ++cref;                        // Error: read-only!
   const string str = "I am a constant string!";
// str = "That doesn't work!";  // Error: str constant!
// string& text = str;          // Error: str constant!
   const string& text = str;    // ok!
   cout << text << endl;        // ok! Just reading.
   return 0;
}
```

REFERENCE

Pointer

- A pointer can be initialized to any value anytime after it is declared.

```
int a = 5;
// some code
int *p = &a;
```

- A pointer can be assigned to point to a *NULL* value.
- Pointers need to be dereferenced with a `*`.
- A pointer can be changed to point to any variable of the same type.

  Example:

```
int a = 5;
int *p;
p = &a;
int b = 6;
p = &b;
```

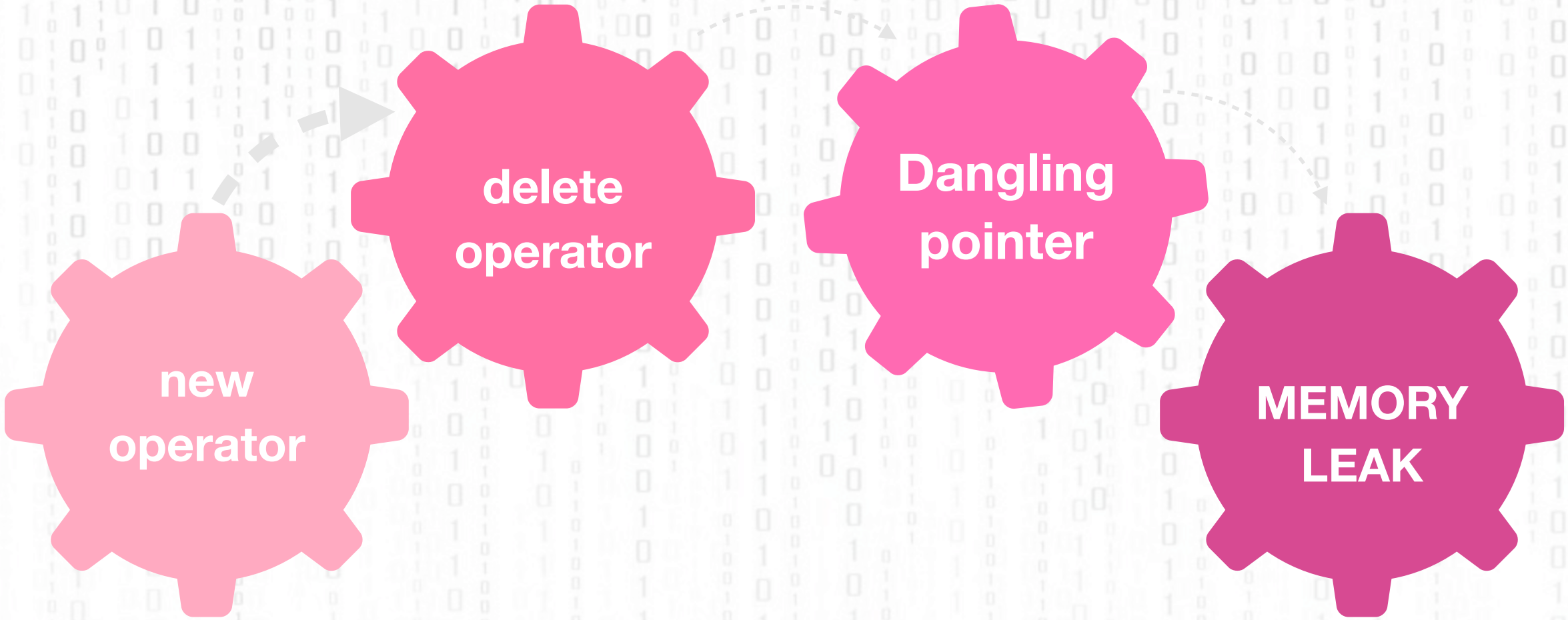
Reference

- A reference must be initialized when it is declared.

```
int a = 5;
int &ref = a;
```

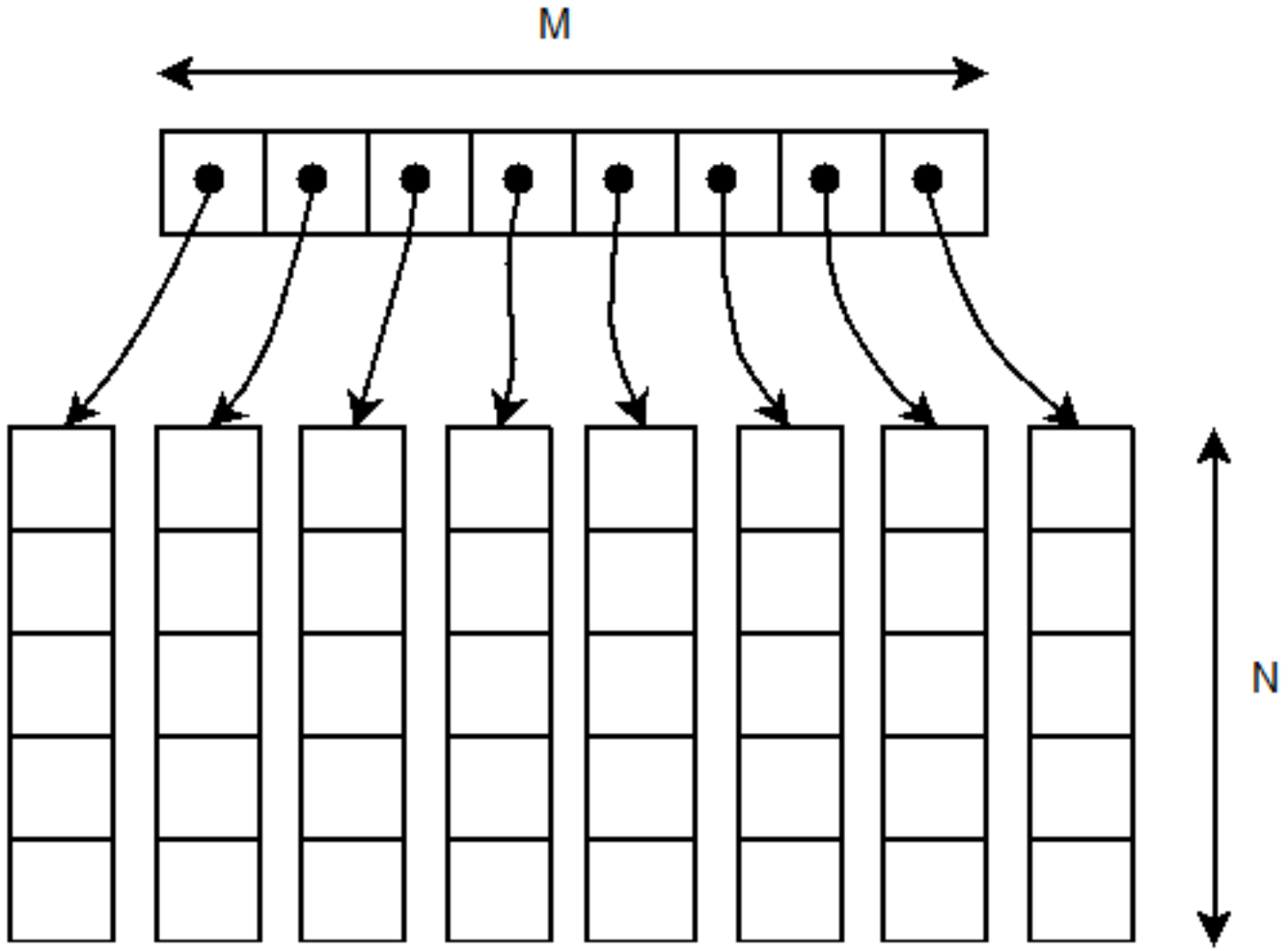
- References cannot be *NULL*.
- References can be used ,simply, by name.
- Once a reference is initialized to a variable, it cannot be changed to refer to a variable object.

https://www.educative.io/edpresso/differences-between-pointers-and-references-in-cpp



DMA

```cpp
#include <iostream>

#define N 10

// Dynamically allocate memory for 1D Array in C++
int main()
{
    // dynamically allocate memory of size `N`
    int* A = new int[N];

    // assign values to the allocated memory
    for (int i = 0; i < N; i++) {
        A[i] = i + 1;
    }

    // print the 1D array
    for (int i = 0; i < N; i++) {
        std::cout << A[i] << " ";     // or *(A + i)
    }

    // deallocate memory
    delete[] A;

    return 0;
}
```



```cpp
#include <iostream>

// `M × N` matrix
#define M 4
#define N 5

// Dynamic Memory Allocation in C++ for 2D Array
int main()
{
    // dynamically create an array of pointers of size `M`
    int** A = new int*[M];

    // dynamically allocate memory of size `N` for each row
    for (int i = 0; i < M; i++) {
        A[i] = new int[N];
    }

    // assign values to the allocated memory
    for (int i = 0; i < M; i++)
    {
        for (int j = 0; j < N; j++) {
            A[i][j] = rand() % 100;
        }
    }
```

```
    // print the 2D array
    for (int i = 0; i < M; i++)
    {
        for (int j = 0; j < N; j++) {
            std::cout << A[i][j] << " ";
        }
        std::cout << std::endl;
    }

    // deallocate memory using the delete operator
    for (int i = 0; i < M; i++) {
        delete[] A[i];
    }
    delete[] A;

    return 0;
}
```

File

A file is a collection of related data stored in a particular area on the disk. The data is stored in disk using the concept of file .

Why File

Permanent storage of data : - (all the message or value printed with help of any output statements like printf , putchar are never available for future use) .

If there is a large amount of data generated as output by a program, storing that output in file will help in easy handling /analysis of the output , as user can see the whole output at any time even after complete execution of the program.

If we need lot of data to be inputted, user cannot keep on typing that again and again for repeated execution of program. In that case, all input data can be once written in a file and then that file can be easily used as the input file.

The transfer of input – data or output – data from one computer to another can be easily done by using files.

Using Input/Output Files

A computer file

- is stored on a secondary storage device (e.g., disk);
- is permanent;
- can be used to
  - provide input data to a program
  - or receive output data from a program
  - or both;
- should reside in Project directory for easy access;
- must be opened before it is used.

General File I/O Steps

1. Include the header file fstream in the program.
2. Declare file stream variables.
3. Associate the file stream variables with the input/output sources.
4. Open the file
5. Use the file stream variables with >>, <<, or other input/output functions.
6. Close the file.

Using Input/Output Files

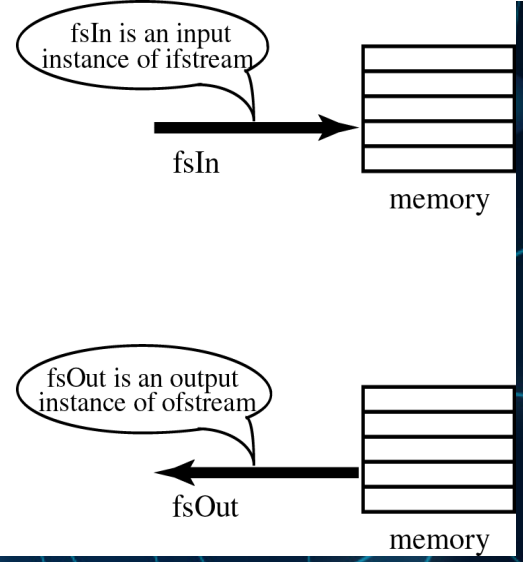
- ***stream*** - a sequence of characters
  - interactive (iostream)
    - **cin** - input stream associated with **keyboard.**
    - **cout** - output stream associated with **display**
  - file (fstream)
    - **ifstream** - defines new input stream (normally associated with a file).
    - **ofstream** - defines new output stream (normally associated with a file).

Stream I/O Library Header Files

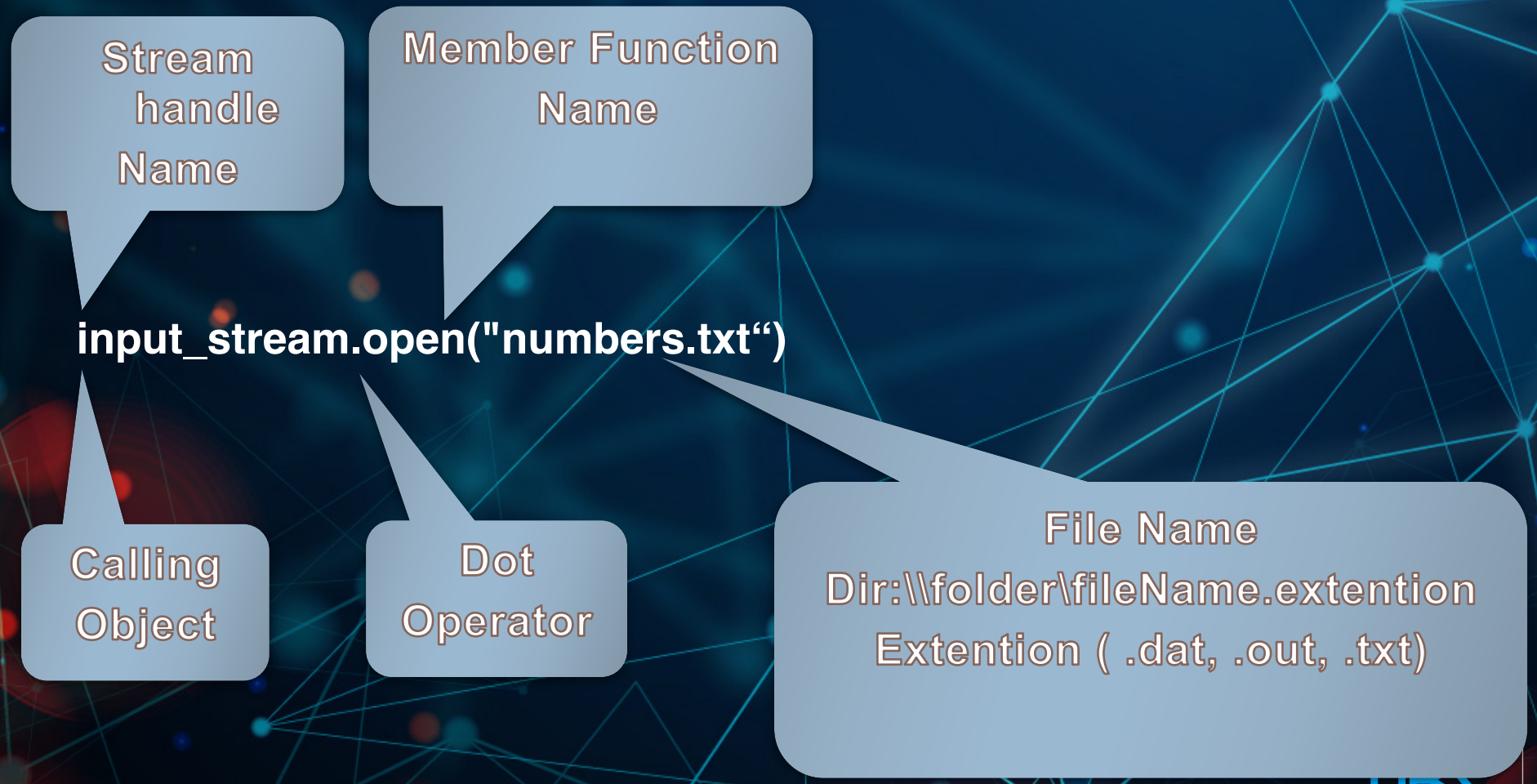
- Note: There is no ".h" on standard header files : <fstream>
- iostream -- contains basic information required for all stream I/O operations
- fstream -- contains information for performing file I/O operations

```
//Add additional header files you use
#include <fstream>
int main ()
{ /* Declare file stream variables such as
 the following  */
    ifstream  fsIn;//input
    ofstream fsOut; // output
    fstream both //input & output
//Open the files
    fsIn.open("prog1.txt"); //open the input file
    fsOut.open("prog2.txt"); //open the output file
//Code for data manipulation
.
.
//Close files
    fsIn.close();
    fsOut.close();
    return 0;
}
```

```
#include <fstream.h>

int main (void)
{
// Local Declarations
   ifstream      fsIn;

   ofstream      fsOut;
   •
   •
   •

}  // main
```



Object and Member Functions



Open()

- Opening a file associates a file stream variable declared in the program with a physical file at the source, such as a disk.
- In the case of an input file:
  - the file must exist before the open statement executes.
  - If the file does not exist, the open statement fails and the input stream enters the fail state
- An output file does not have to exist before it is opened;
  - if the output file does not exist, the computer prepares an empty file for output.
  - If the designated output file already exists, by default, the old contents are erased when the file is opened.

Validate the file before trying to access

By checking the stream variable;

```
If ( ! Mystream)
{
Cout << “Cannot open file.\n ”;
}
```

By using bool is_open() function.

```
If ( ! Mystream.is_open()) {
Cout << “File is not open.\n ”;
}
```

File I/O Example: Open the file with validation

```
#include <fstream>
using namespace std;
int main()
{
//declare output file variable
    ofstream outFile;
// open an exist file fout.txt
    outFile.open("fout.txt");
// Open validation
    if(! outFile.is_open() ) {
            Cout << "Cannot open file.\n ";
            return 1;
    }
    return 0;
}
```

File I/O Example: Open the file with validation

```
#include <fstream>
using namespace std;
int main()
{
//declare and automatically open the file
    ofstream outFile(“fout.txt");
// Open validation
    if(! outFile) {
        Cout << “Cannot open file.\n ”;
        return 1;
    }
    return 0;
}
```

More Input File-Related Functions

- **ifstream fsin;**
- **fsin.open(const char[] fname)**
  - connects stream **fsin** to the external file ***fname.***
- **fsin.get(char character)**
  - extracts next character from the input stream **fsin** and places it in the character variable ***character.***
- **fsin.eof()**
  - tests for the end-of-file condition.

File I/O Example: Reading

```
#include <iostream>
#include <fstream>

int main()
{//Declare and open a text file
    ifstream openFile("data.txt");
    char ch;
 //do until the end of file
    while( ! openFile.eof() )
    {
        openFile.get(ch); // get one character
        cout << ch;   // display the character
    }
    openFile.close(); // close the file

    return 0;
}
```

File I/O Example: Reading

```
#include <iostream>
#include <fstream>
#include <string>
int main()
{//Declare and open a text file
      ifstream openFile("data.txt");
      string line;
      while(!openFile.eof())
      {//fetch line from data.txt and put it in a string
            getline(openFile, line);
            cout << line;
      }
      openFile.close(); // close the file
      return 0;
}
```

More Output File-Related Functions

- **ofstream fsOut;**
- **fsOut.open(const char[] fname)**
  - connects stream **fsOut** to the external file ***fname.***
- **fsOut.put(char character)**
  - inserts character ***character*** to the output stream **fsOut.**
- **fsOut.eof()**
  - tests for the end-of-file condition.

File I/O Example: Writing

```
#include <fstream>
using namespace std;
int main()
{// declare output file variable
     ofstream outFile;
// open an exist file fout.txt
    outFile.open("fout.txt”);
//behave just like cout, put the word into  the
file
      outFile << "Hello World!";
      outFile.close();
       return 0;
}
```

File I/O Example: Writing

```
#include <fstream>
using namespace std;
int main()
{/* declare and automatically open the
file*/

     ofstream outFile("fout.txt");
//behave just like cout, put the word into
the file

     outFile << "Hello World!";
     outFile.close();


return 0;
}
```