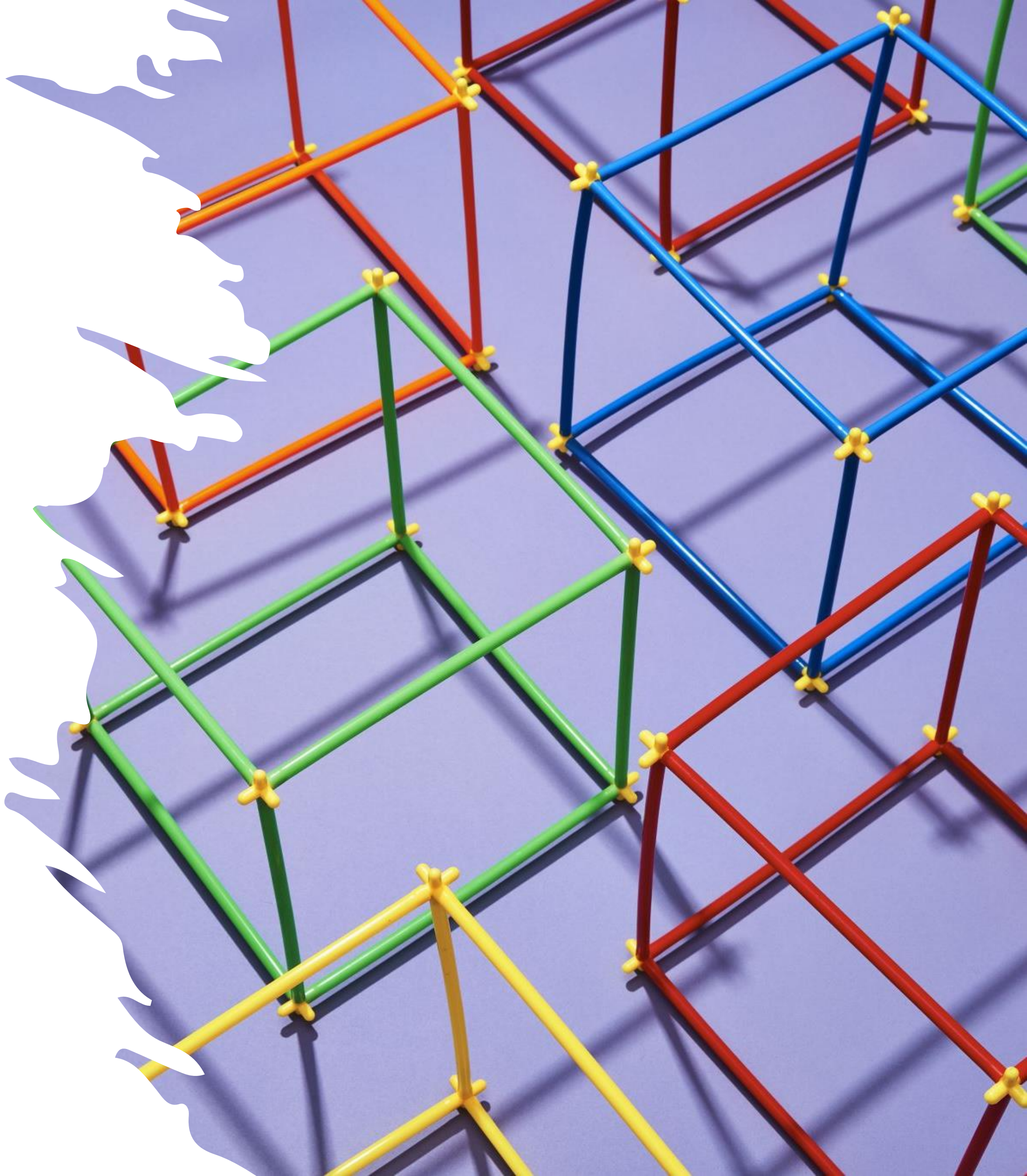
Data Structure Lecture 3: Arrays and Pointers

Prepared by

Dr. Mohammed Salah Al-Obiadi



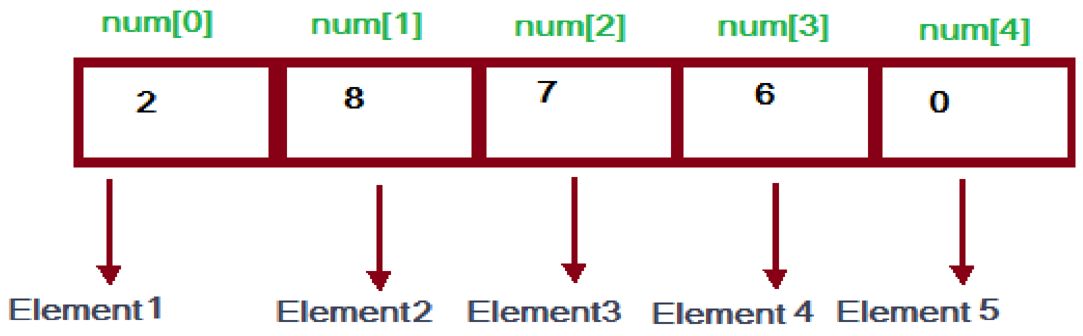
Arrays data structures

- Arrays are widely used in any programming language.
- It is extremely useful in cases where we need to store the similar set of elements.
- It helps in reducing the program complexity.
- increases the programmer’s productivity.
- Arrays can be categorized into the following:
  - Single Dimensional array.
  - Double Dimensional array.
  - Multidimensional array.
- We will not study the Multidimensional array.

Why we use Arrays

- Consider that we need to store grades of five students.
- In a normal way, we have to define five variables of the same type:

```
int main ()
  {
      int marks1, marks2, marks3, marks4, marks5;
      cout<<"enter marks1";
      cin>>marks1;
      cout<<"enter marks2";
      cin>>marks2;
      cout<<"enter marks3";
      cin>>marks3;
      cout<<"enter marks4";
      cin>>marks4;
      cout<<"enter marks5";
      cin>>marks5;
      return 0;
  }
```



- Complexity of the above program will grow further upon increment of subjects.
- Consider we have 200 students, how the program will look like? What is the solution?
- Here the solutions lie with the usage of arrays.

Array can be defined as:

- A data structure used to store set of similar data types.
- Elements are stored in continuous memory locations.
- Index, or subscript starts with 0.
- Size of the array should be constant.

One-Dimensional Array

Declaration:

- *Data type variable_name[bound] ;*

Examples:

- *Int arr[10]; // an integer array with 10 elements.*
- *Char arr[20]; // a character array with 20 elements.*
- *float arr[15]; // a flaot array with 20 elements*

Array Element in Memory

The array elements are stored in a consecutive manner inside the memory.

For Example: int x[7];

Let the x[0] be at the memory address 568, then the entire array can be represented in the memory as:

x[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]
568	570	572	574	576	578	580

Two-Dimensional Array

Declaration:

- *Data type variable_name[rows] [columns] ;*

Examples:

- *Int arr[4][6]; // an integer 2-D array with 4 rows and 6 columns.*
- *Char arr[20][20]; // a character 2-D array with 20 rows and 20 columns.*
- *float arr[5][10]; // a float 2-D array with 5 rows and 10 columns*

Examples of Two-dimensional arrays

```
int x[3][4]={
                        {1, 2, 3, 4},
                        {5, 6, 7, 8},
                        {2, 4, 6, 3},
                     };
char x[3][4]={
                        {'h', 'a', 'f', '7'},
                        {'u', 'f', 'z', 'l'},
                        {'y', '8', 'j', 'm'},
                     };
```

POINTERS

Pointer is a variable that is capable to hold the address of another variable.

Holding of addresses of another variable is needed in various instances that include:

- 1- To access the array element
- 2- To change the value of variable from function
- 3- In dynamic allocation of memory.
- 4- In complex programming, such as link list, tree, B tree etc.

How to know a variable is a pointer?

Pointers are preceded with the symbol *.

For instance:

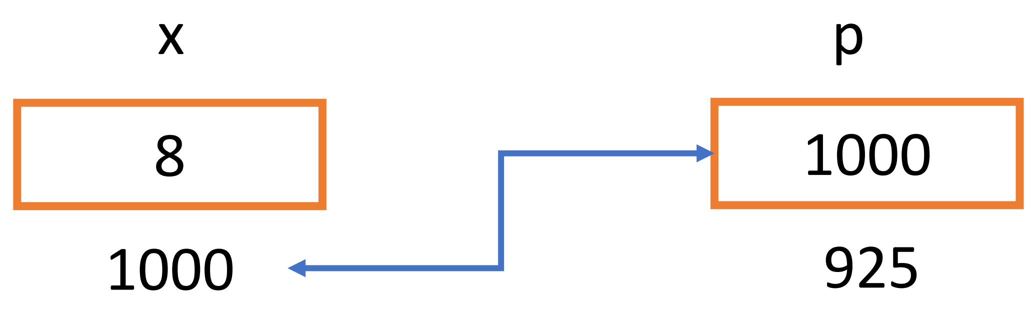
- int *x, It means that this pointer can hold the address of integer type variable.
- char *c, , It means that this pointer can hold the address of char type variable.
- float *w, It means that this pointer can hold the address of float type variable.

Example of Declaring pointer

```
int x=8;
int *p;// variable that is pointer of int type
p=&x; //p now holds the address of variable x
cout<<p; // print the address of x;
cout<<*p; // print the value pointed by p;
```

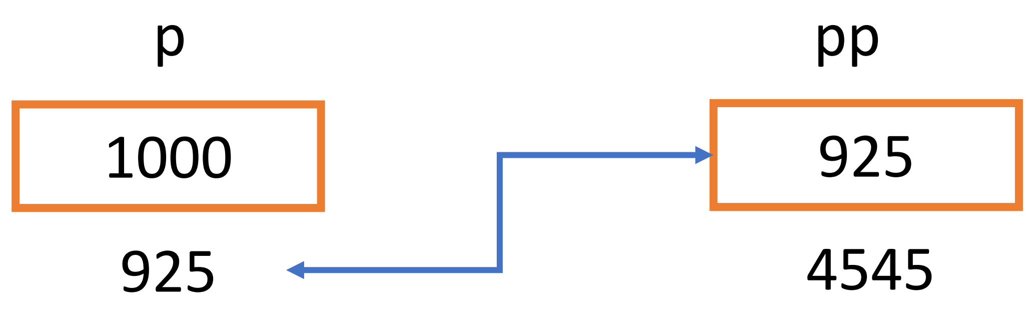
Explaining Example of Declaring pointer

- Initially, the variable "x" is declared
- Assumes that it has been allocated the address location 1000.
- when int *p is declared, it is also allocated the address 925.
- When p=&x, this means that p holds the address of variable x which is 1000.
- Printing p will print address while printing *p will print x value.



Pointer to pointer

- Sometimes, we need to store the address of a pointer.
- This can be accomplished with the help of pointer to pointer.
- Pointer to pointer is a variable that holds the address of another variable that is pointer type.
- Declaring pointer to pointer is different from the normal pointer type.
- In pointer to pointer notation two asterisk (**) are preceded before the identifier.
- Example:-
  - int **pp;
  - int *p;
  - pp=&p;


p
1000
925
pp
925
4545