

Data Structures

Lecture 1

Prepared By: Prof Sweta Dargad

LECTURE 1

Course Information

► Textbooks

- 1. An Introduction to Data Structures with Application, By Jean-Paul Tremblay ,Paul G. Sorenson (second Edition)

► Reference Books

- 1. Data Structures using C & C++1 -By Ten Baum Publisher – Prentice-Hall International.
- 2. Fundamentals of Computer Algorithms by Horowitz, Sahni, Galgotia Pub. 2001 ed.
- 3. Fundamentals of Data Structures in C++-By Sartaj Sahani.

Course Outline

- ▶ Introduction to Data Structure
- ▶ Algorithms
- ▶ Recursion
- ▶ Stacks
- ▶ Queues
- ▶ Lists and linked lists
- ▶ Trees
- ▶ Sorting
- ▶ Searching
- ▶ Graphs
- ▶ Hashing

Grading

▶ Theory

- ▶ Quizzes -----10%
- ▶ Assignments and Attendance-----10%
- ▶ Mid Term----- 20%
- ▶ Final----- 60%

▶ Labs

- ▶ Assignments/Exercises and Project (Internal)----- 40%
- ▶ External----- 60%

Why do we need a Data Structure??

7/24/2020

- ▶ **Data** are simply sets of Values

Ex: 24, Rajesh, 23.5,

- ▶ A *Data items* refers to single unit of values

Ex: 24

- ▶ Data items that are divided into sub items are called *group items*;

- ▶ those that are not are called *elementary items*

Ex: Employee name & SSN Number

```
graph TD; A[Data items] --- B[group items]; A --- C[elementary items]
```

Data items

*group
items*

*elementary
items*

7/24/2020

- ▶ Employee name (*group items*)

Ex: Mahesh Kumar Gupta

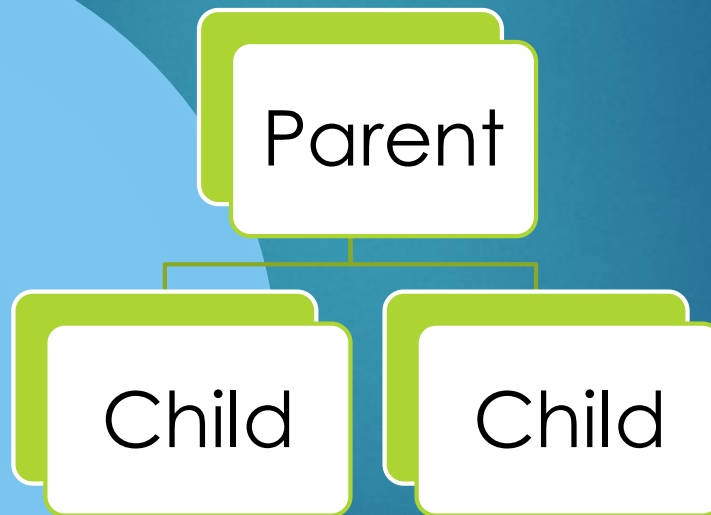
- ▶ SSN Number (*elementary items*)

Ex: 2344-5456-6756

9754679843

- **Collections** of data frequently organized into hierarchy of *fields, records & files*

7/24/2020



files

12



Filed 1



Filed 2



Field 3

Untitled - Notepad			
File Edit Format View Help			
Record 1	→	Mahesh Kumar Gupta	2344-5456-6756 8754679845
Record 2	→	Ram Praksh	4344-5456-6757 5754679847
Record 3	→	Rakesh Kuamr Pal	6344-5456-6758 9854679846
Record 4	→	Yatharth	7344-5456-6759 7754679843

Additional terminology

7/24/2020

- ▶ An *entity* is something that has certain *attributes* or properties which may be assigned values
- ▶ **Values** themselves either numeric or non numeric

**Entity
set**

Record 1 ➡

Record 2 ➡

Record 3 ➡

Record 4 ➡

**Attribute 1
(Name)**

**Attribute 2
(\$\$N No.)**

**Attribute 3
Mobile No.**

Mahesh Kumar Gupta	2344-5456-6756	8754679845
Ram Praksh	4344-5456-6757	5754679847
Rakesh Kuamr Pal	6344-5456-6758	9854679846
Yatharth	7344-5456-6759	7754679843

- ▶ Entities with similar attributes form an entity set.
- ▶ Each attribute of an entity set has a *range* of values, the set of all possible values that could be assigned to the particular attribute.

- ▶ The term “information” is some times used for data with given attributes, or in other words meaningful or process data
- ▶ The way that data are organized into hierarchy of *fields , records & files* reflects the **relationship** between attributes , entity & entity sets

▶ A *field* is single elementary unit of information representing an attributes of an entity.

▶ A *record* is collection of field values of a given entity

▶ A *file* is the collection records of the entities in a given entity set

Each record in a file may contain many field items, but the value in a certain field may uniquely determine the record in the file. Such a field K is called a *primary key*, and the values k_1, k_2, \dots in such a field are called *keys* or *key values*.

- (a) Suppose an automobile dealership maintains an inventory file where each record contains the following data:

Serial Number, Type, Year, Price, Accessories

The Serial Number field can serve as a primary key for the file, since each automobile has a unique serial number.

- (b) Suppose an organization maintains a membership file where each record contains the following data:

Name, Address, Telephone Number, Dues Owed

Although there are four data items, Name and Address may be group items. Here the Name field is a primary key. Note that the Address and Telephone Number fields may not serve as primary keys, since some members may belong to the same family and have the same address and telephone number.

- ...
- ▶ Records may also be classified according to length
 - ▶ A file can have a fixed length records or variable length records
 - ▶ In *fixed length records* all the records contains the same data items with the same amount of space assigned to each data items
 - ▶ In variable *length records* file records may contain different lengths

For example, student records usually have variable lengths, since different students take different numbers of courses. Usually, variable-length records have a minimum and a maximum length.

The above organization of data into fields, records and files may not be complex enough to maintain and efficiently process certain collections of data. For this reason, data are also organized into more complex types of structures. The study of such data structures, which forms the subject matter of this text, includes the following three steps:

1. Logical or mathematical description of the structure
2. Implementation of the structure on a computer
3. Quantitative analysis of the structure, which includes determining the amount of memory needed to store the structure and the time required to process the structure.

Remark: The second and third of the steps in the study of data structures depend on whether the data are stored (a) in the main (primary) memory of the computer or (b) in a secondary (external) storage unit. This text will mainly cover the first case. This means that, given the address of a memory location, the time required to access the content of the memory cell does not depend on the particular cell or upon the previous cell accessed. The second case, called *file management* or *data base management*, is a subject unto itself and lies beyond the scope of this text.

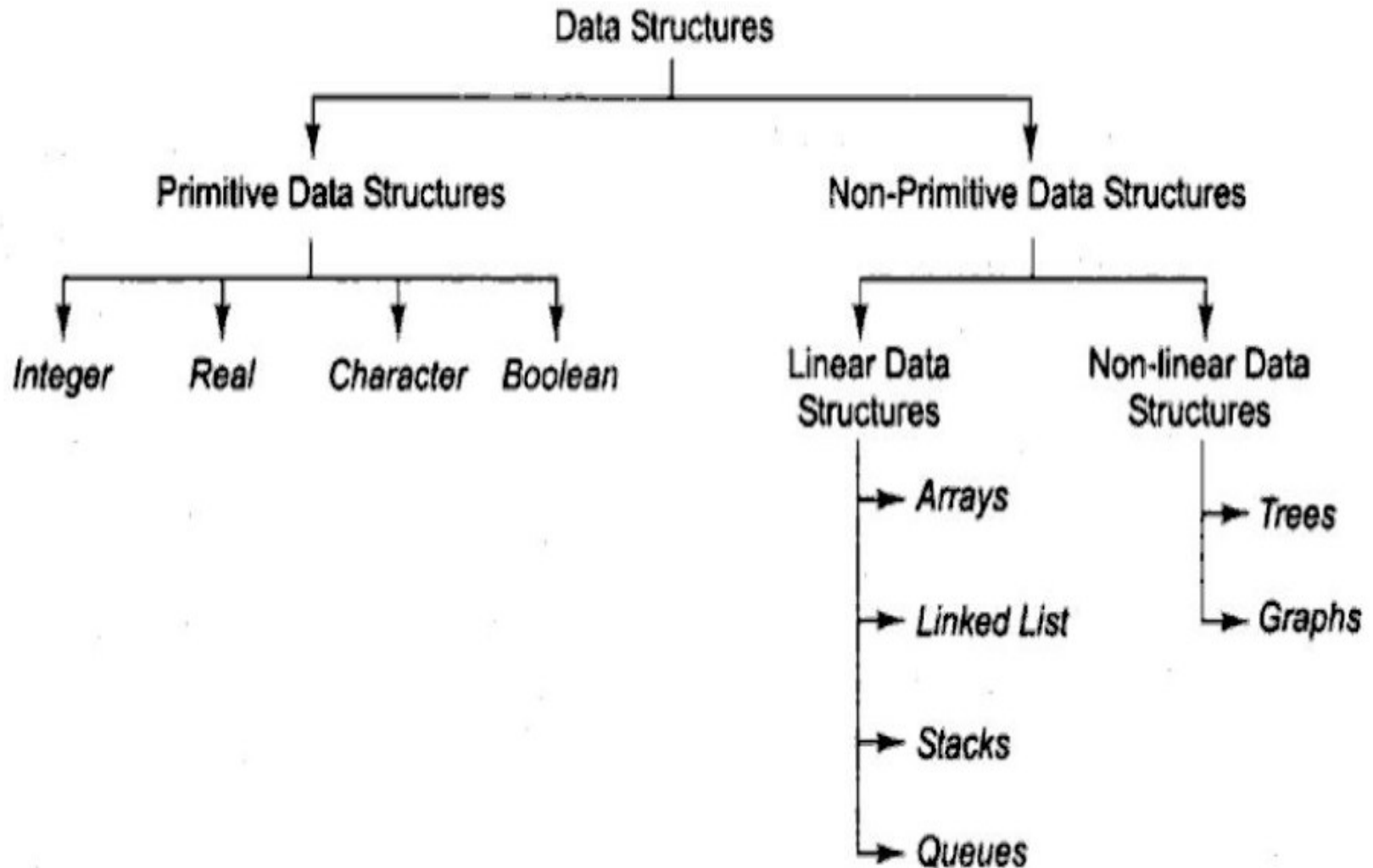
The background is a dark blue gradient. It features several light blue circles of varying sizes. A small green rectangle is located in the top right corner. The text is white and centered.

Introduction to Data Structure and Abstract Data Types

What is Data Structure?

- ▶ Data structure is a representation of data and the operations allowed on that data.
- A data structure is a way to store and organize data in order to facilitate the access and modifications.
- Data Structure are the method of representing of logical relationships between individual data elements related to the solution of a given problem.

Classification



Primitive Data Types

- ▶ These data Types consist of characters that can not be divided
- ▶ Hence they are also called simple data types

7/24/2020

Non primitive data types

7/24/2020

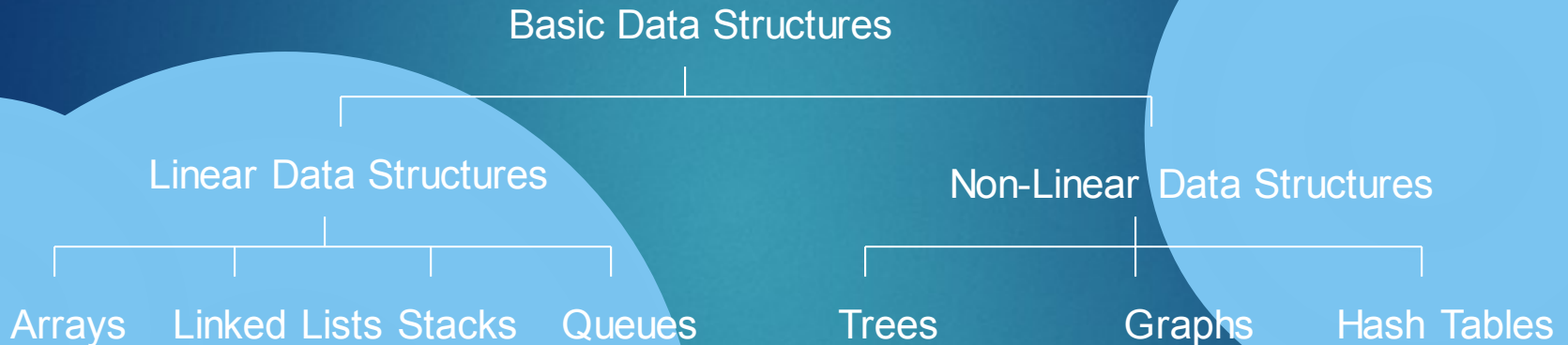
- ▶ A data structure is said to be *linear* if its elements form a sequence or a linear list
- ▶ In linear data structure data is arranged in a linear fashion, the way they are stored in memory need not be sequential
- ▶ Ex. Arrays, linked list, stacks & queues

...

- ▶ A data structure is said to be *non linear* if the data is not arranged in sequence
- ▶ The insertion & deletion of data is therefore not possible in a linear fashion
- ▶ Ex. Trees & graphs

7/24/2020

Basic Data Structure

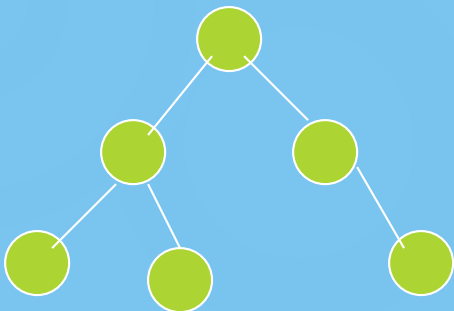




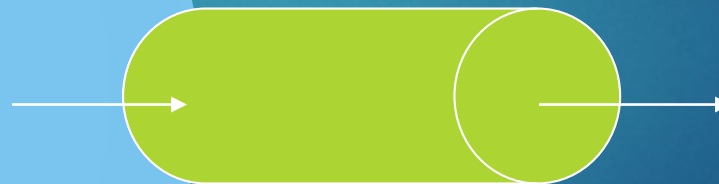
array



Linked list



tree



queue



stack

Selection of Data Structure

- ▶ The choice of particular data model depends on two consideration:
 - ▶ It must be rich enough in structure to represent the relationship between data elements
 - ▶ The structure should be simple enough that one can effectively process the data when necessary

Types of Data Structure

- ▶ Linear: In Linear data structure, values are arranged in linear fashion.
 - ▶ Array: Fixed-size
 - ▶ Linked-list: Variable-size
 - ▶ Stack: Add to top and remove from top
 - ▶ Queue: Add to back and remove from front
 - ▶ Priority queue: Add anywhere, remove the highest priority

Types of Data Structure

- ▶ Non-Linear: The data values in this structure are not arranged in order.
 - ▶ Hash tables: Unordered lists which use a 'hash function' to insert and search
 - ▶ Tree: Data is organized in branches.
 - ▶ Graph: A more general branching structure, with less strict connection conditions than for a tree

Type of Data Structures

- ▶ Homogenous: In this type of data structures, values of the same types of data are stored.

- ▶ Array

- ▶ Non-Homogenous: In this type of data structures, data values of different types are grouped and stored.

- ▶ Structures

- ▶ Classes

Abstract Data Type and Data Structure

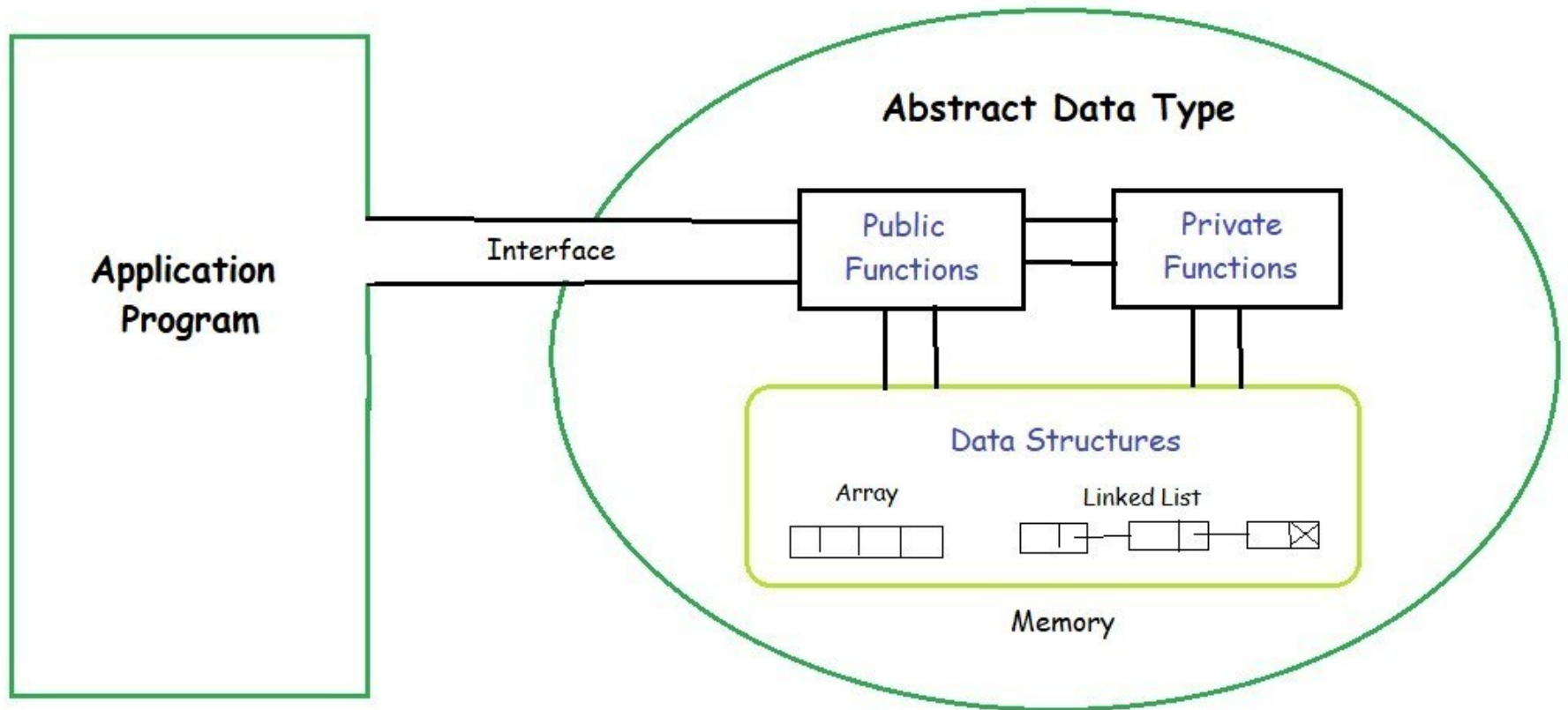
► Definition:-

- *Abstract Data Types (ADTs)* stores data and allow various operations on the data to access and change it.
- A mathematical model, together with various operations defined on the model
- An ADT is a collection of data and associated operations for manipulating that data

► Data Structures

- Physical implementation of an ADT
- data structures used in implementations are provided in a language (*primitive* or *built-in*) or are built from the language constructs (*user-defined*)
- Each operation associated with the ADT is implemented by one or more subroutines in the implementation

ADT



Abstract Data Type

- ▶ ADTs support *abstraction*, *encapsulation*, and *information hiding*.
- ▶ *Abstraction* is the structuring of a problem into well-defined entities by defining their data and operations.
- ▶ The principle of hiding the used data structure and to only provide a well-defined interface is known as *encapsulation*.

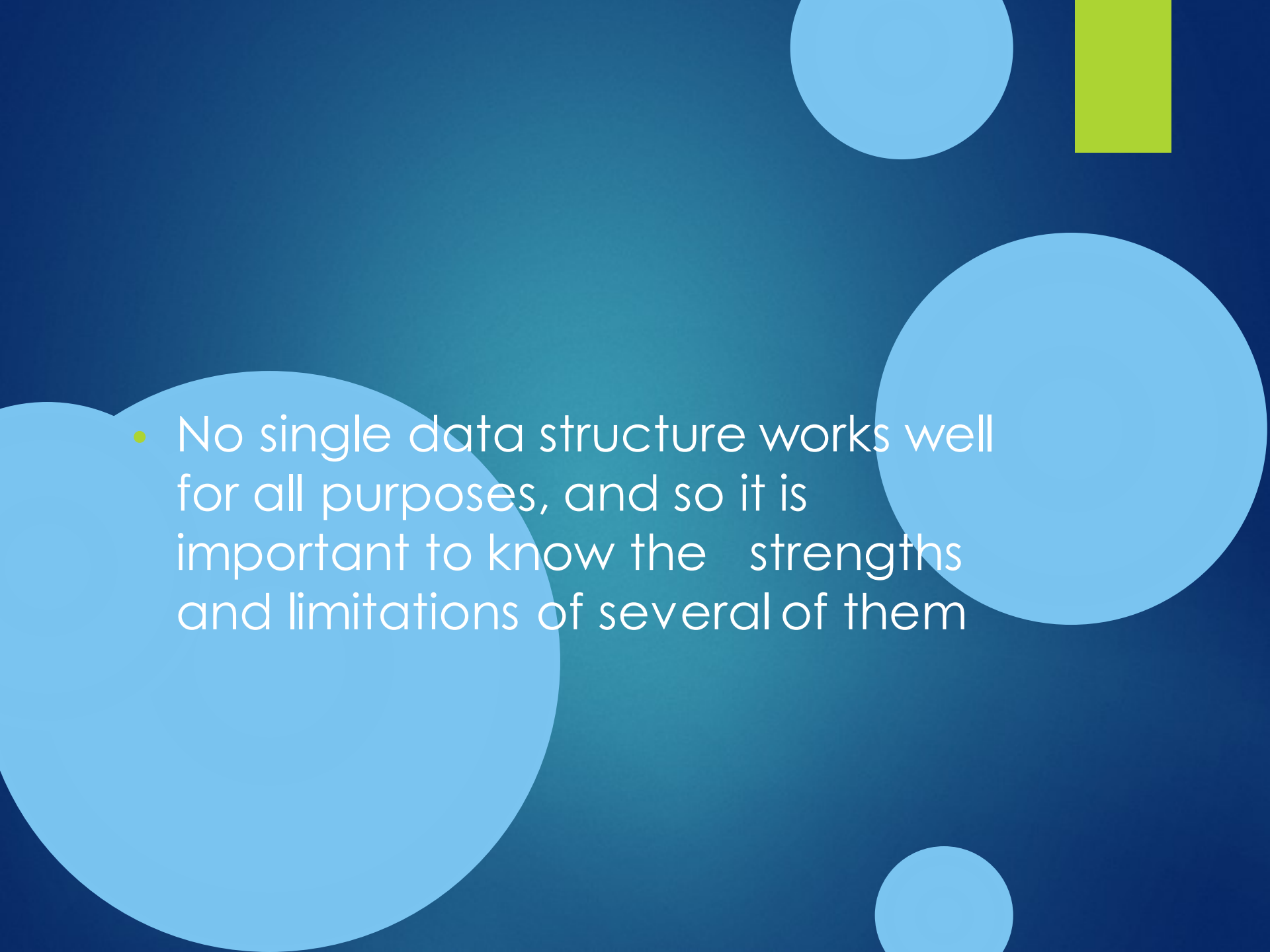
The Core Operations of ADT

- ▶ Every Collection ADT should provide a way to:

- ▶ add an item
- ▶ remove an item
- ▶ find, retrieve, or access an item

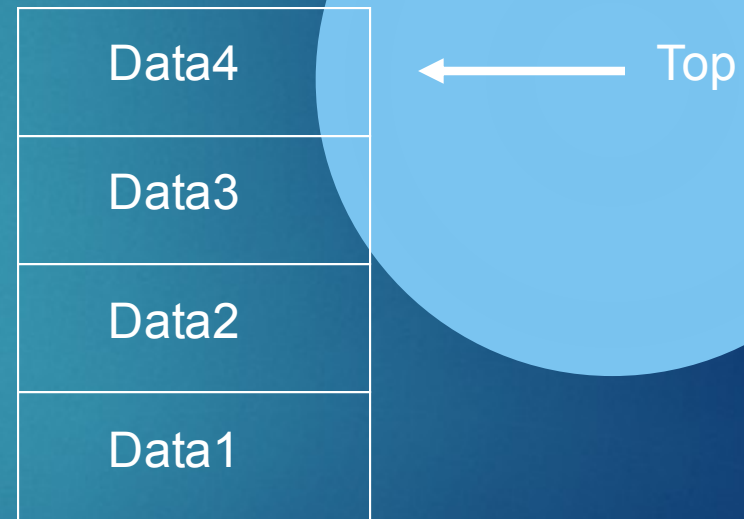
- ▶ Many, many more possibilities

- ▶ is the collection empty
- ▶ make the collection empty
- ▶ give me a sub set of the collection

- 
- The background is a dark blue gradient. It features several light blue circles of different sizes: a large one on the left, a medium one on the right, a small one at the top center, and another small one at the bottom center. A vertical green rectangle is located in the top right corner.
- No single data structure works well for all purposes, and so it is important to know the strengths and limitations of several of them

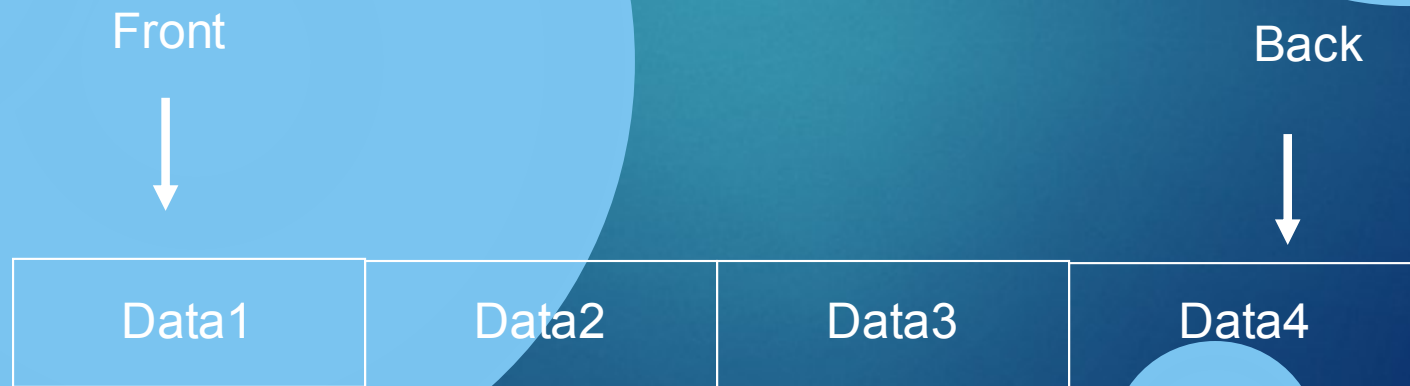
Stacks

- ▶ Collection with access only to the last element inserted
- ▶ Last in first out
 - ▶ insert/push
 - ▶ remove/pop
 - ▶ top
 - ▶ make empty



Queues

- ▶ Collection with access only to the item that has been present the longest
- ▶ Last in last out or first in first out
- ▶ enqueue, dequeue, front
- ▶ priority queues and dequeue



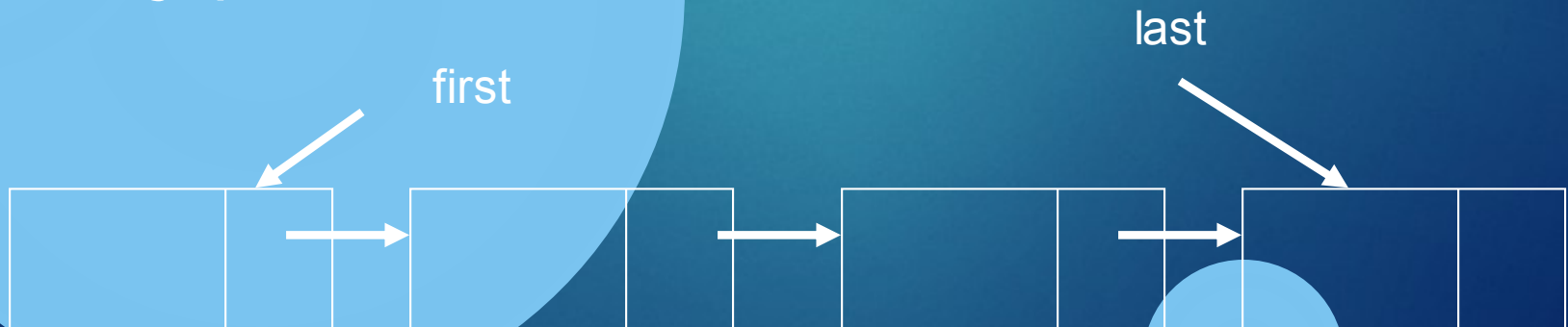
List

- ▶ A ***Flexible*** structure, because can grow and shrink on demand.

Elements can be:

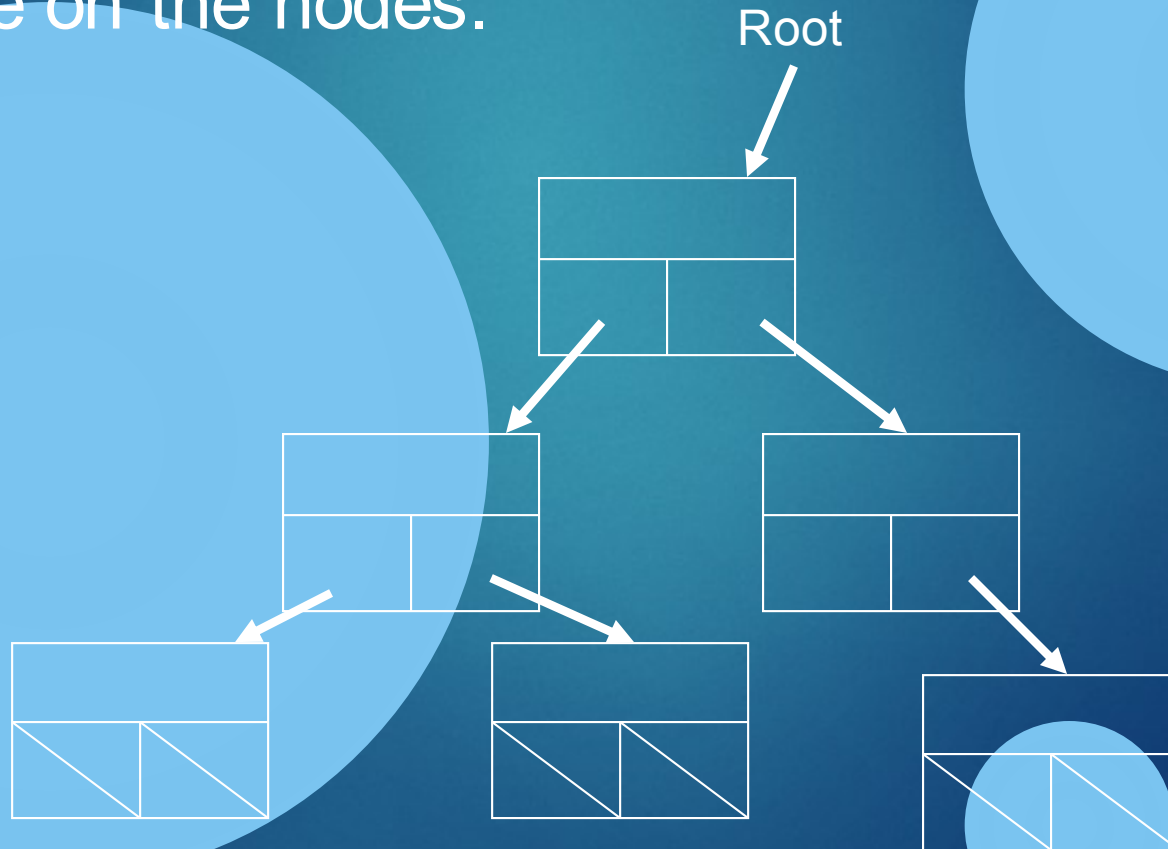
- Inserted
- Accessed
- Deleted

At ***any*** position



Tree

- ▶ A **Tree** is a collection of elements called **nodes**.
- ▶ One of the node is distinguished as a **root**, along with a relation (“parenthood”) that places a hierarchical structure on the nodes.



The background is a dark blue gradient. It features several light blue circles of varying sizes. A small yellow rectangle is located in the top right corner.

Programming with C

Data types in C

- ▶ Data types specify how we enter data into our programs &
- ▶ what type of data we enter
- ▶ C language has some *predefined set of data types* to handle various kinds of data that we can use in our program
- ▶ These datatypes have different storage capacities.

C language supports 2 different type of data types:

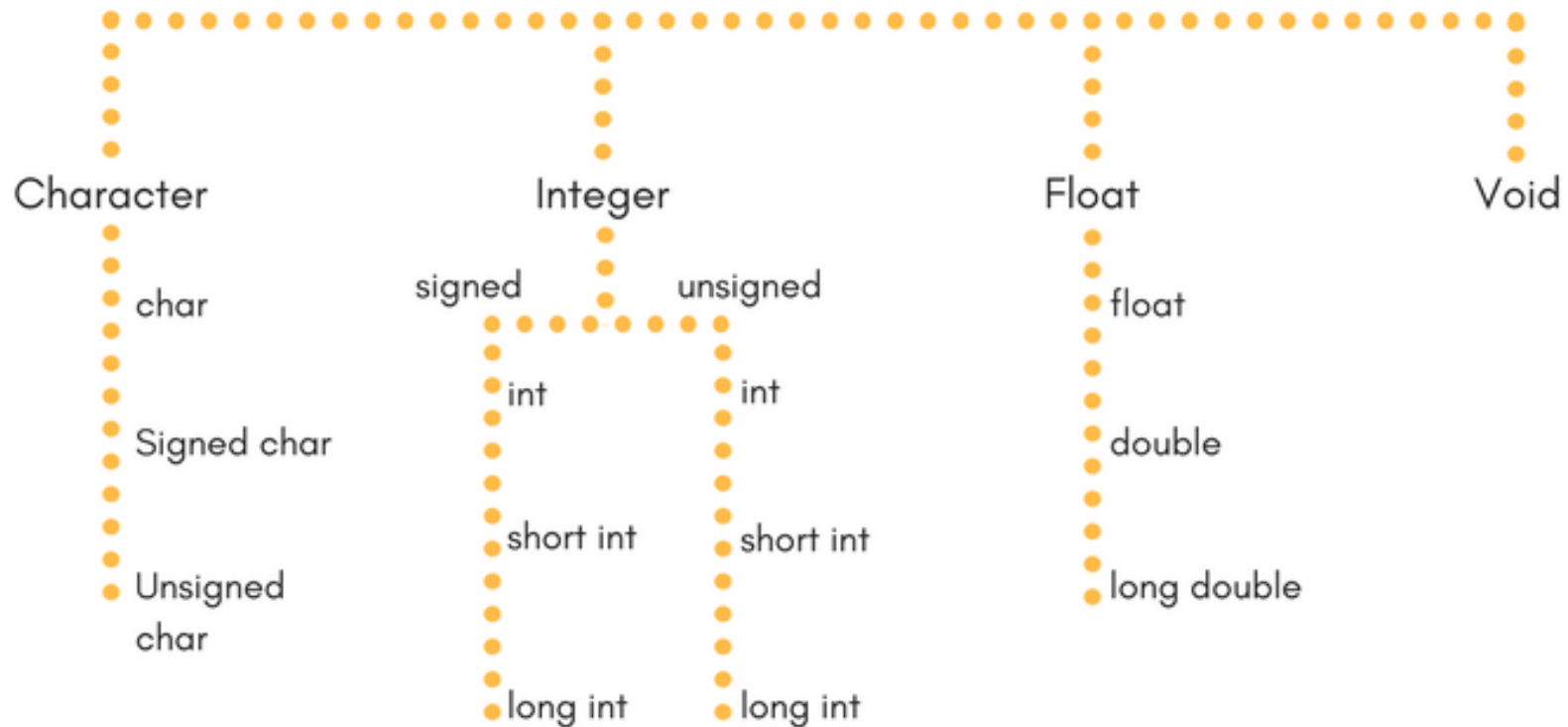
- ▶ **Primary data types:** These are fundamental data types in C namely integer(**int**), floating point(**float**), character(**char**) and **void**.
- ▶ **Derived data types:** Derived data types are nothing but primary datatypes but a little twisted or grouped together like **array**, **structure**, **union** and **pointer**. These are discussed in details later.

...

7/24/2020

- ▶ Data type determines the type of data a variable will hold
- ▶ If a variable `x` is declared as `int`, it means `x` can hold only integer values
- ▶ Every variable which is used in the program must be declared as what data-type it is

Primary Data Type



Integer type

- ▶ Integers are used to store whole numbers.
- ▶ **Size and range of Integer type on 16-bit machine:**

7/24/2020

Type	Size(bytes)	Range
int or signed int	2	-32,768 to 32767
unsigned int	2	0 to 65535
short int or signed short int	1	-128 to 127
unsigned short int	1	0 to 255
long int or signed long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295

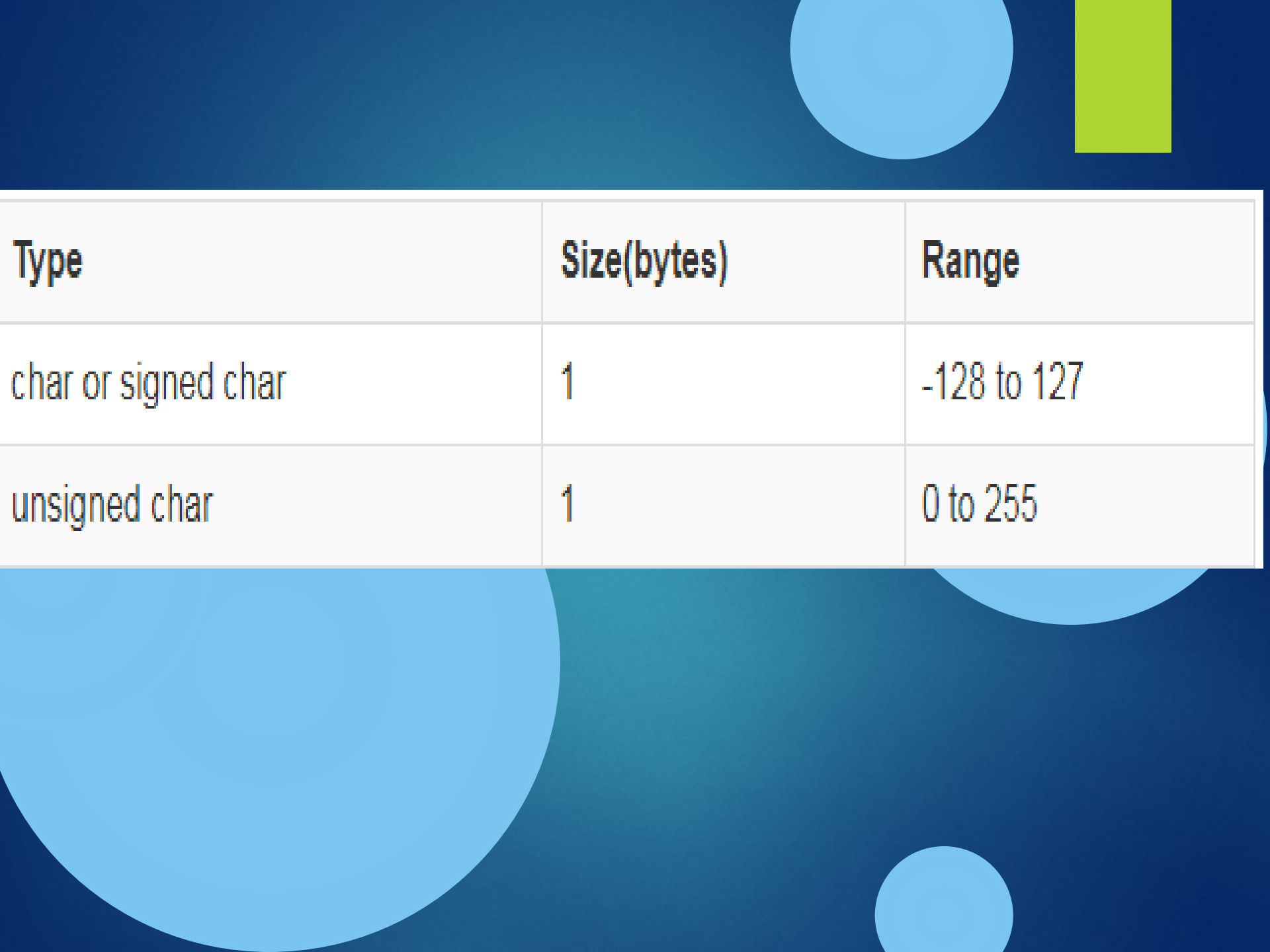
Floating point type

- ▶ Floating types are used to store real numbers.
- ▶ **Size and range of Integer type on 16-bit machine**

Type	Size(bytes)	Range
Float	4	3.4E-38 to 3.4E+38
double	8	1.7E-308 to 1.7E+308
long double	10	3.4E-4932 to 1.1E+4932

Character type

- ▶ Character types are used to store characters value.
- ▶ **Size and range of Integer type on 16-bit machine**



Type	Size(bytes)	Range
char or signed char	1	-128 to 127
unsigned char	1	0 to 255

void type

- ▶ void type means no value
- ▶ this is usually used to specify the type of functions which return nothing

structures in C

7/24/2020

- ▶ A structure is a user-defined data type available in C that allows to combining data items of different kinds
- ▶ Structures are used to represent a record
- ▶ **Defining a structure:** To define a structure, you must use the **struct** statement. The struct statement defines a new data type, with more than one member

....

```
struct [structure name]
{
    member definition;
    member definition; ...
    member definition;
};
```

7/24/2020

....

```
// declaring structure  
struct struct_example  
{  
    int integer;  
    float decimal;  
    char name[20];  
};
```

7/24/2020

union

- ▶ A union is a special data type available in C that allows storing different data types in the same memory location
- ▶ You can define a union with many members, but only one member can contain a value at any given time
- ▶ Unions provide an efficient way of using the same memory location for multiple purposes

...

- ▶ **Defining a Union:** To define a union, you must use the **union** statement in the same way as you did while defining a structure
- ▶ The union statement defines a new data type with more than one member for your program

7/24/2020

...

```
union [union name]
{
  member definition;
  member definition; ...
  member definition;
};
```

7/24/2020

• • • •

// declaring union

```
union union_example
{
    int integer;
    float decimal;
    char name[20];
};
```

7/24/2020

STRUCTURE

UNION

Keyword

The keyword **struct** is used to define a structure

The keyword **union** is used to define a union.

Size

When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is **greater than or equal to the sum of sizes of its members.**

when a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of **union is equal to the size of largest member.**

Memory

Each member within a structure is assigned unique storage area of location.

Memory allocated is shared by individual members of union.

Value Altering

Altering the value of a member will not affect other members of the structure.

Altering the value of any of the member will alter other member values.

Accessing members

Individual member can be accessed at a time.

Only one member can be accessed at a time.

Initialization of Members

Several members of a structure can initialize at once.

Only the first member of a union can be initialized.

Arrays

Arrays are a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

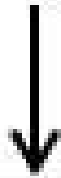
Instead of declaring individual variables, such as `number0`, `number1`, ..., and `number99`, you declare one array variable such as `numbers` and use `numbers[0]`, `numbers[1]`, and ..., `numbers[99]` to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

First Element



Last Element



Numbers[0]

Numbers[1]

Numbers[2]

Numbers[3]

.....

To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows –

```
type arrayName [ arraySize ];  
double balance[10];
```

Here balance is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

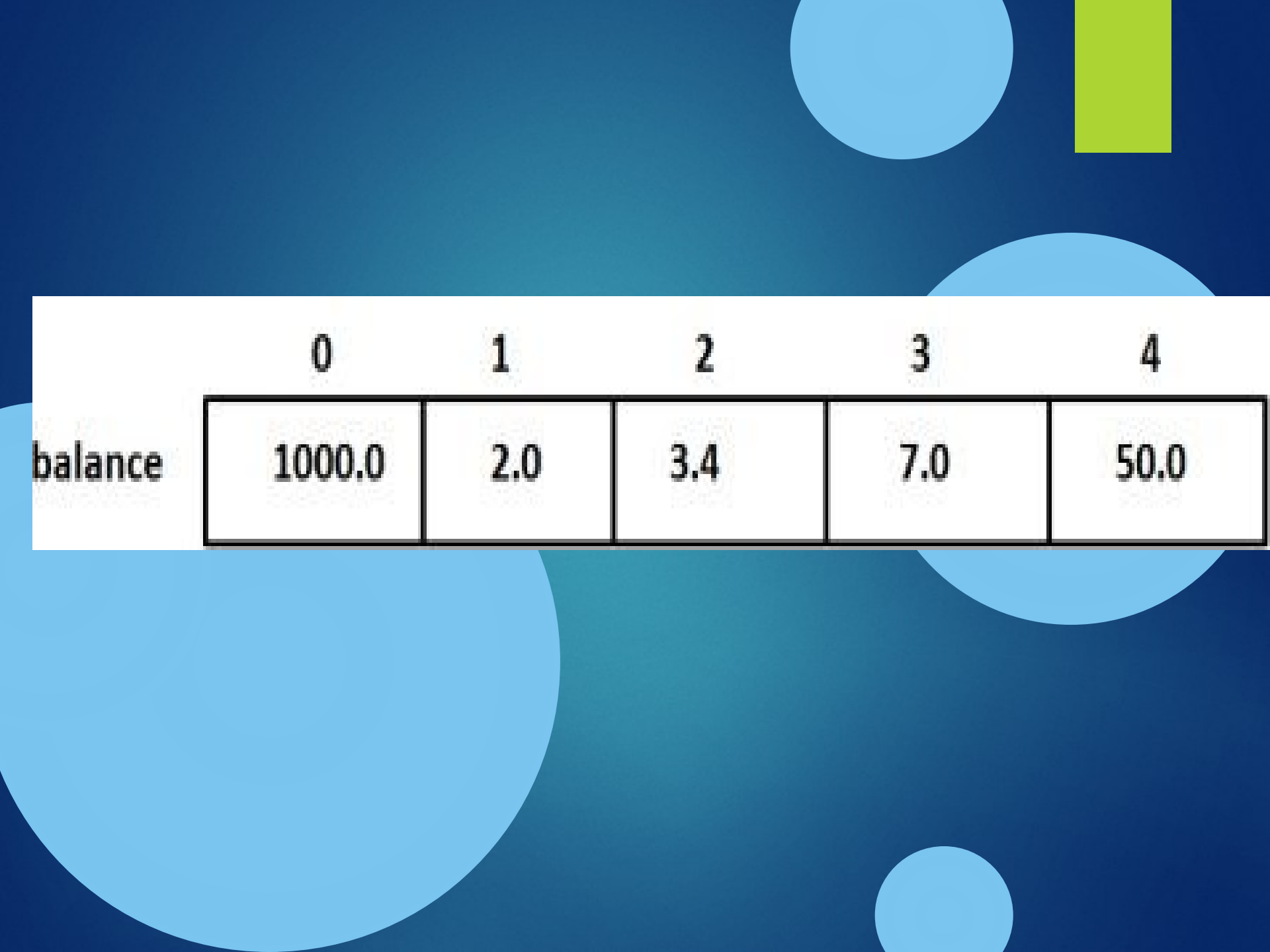
You can initialize an array in C either one by one or using a single statement as follows –

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

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [].

If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write –

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



	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

Recursion.

Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.

```
void recursion() {  
    recursion(); /* function calls itself */  
}
```

Pointers.

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 using it to store any variable address. The general form of a pointer variable declaration is –

```
type *var-name;
```

NULL Pointers

It is always a good practice to assign a NULL value to a pointer variable in case you do not have an exact address to be assigned. This is done at the time of variable declaration. A pointer that is assigned NULL is called a null pointer.

Strings

Strings are actually one-dimensional array of characters terminated by a null character '\0'. Thus a null-terminated string contains the characters that comprise the string followed by a null.

The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

```
char greeting[] = "Hello";
```

Index

0

1

2

3

4

5

Variable

H

e

l

l

o

\0

Address

0x23451

0x23452

0x23453

0x23454

0x23455

0x23456