Programming Document

Sofiullah Iqbal Kiron
sofiul.k.1023@gmail.com

BSMRSTU, Department of CSE
SHIICT

11 April, 2020
4:16pm
Version: 0.0.334

A-Z Menu

List of Figures

List of Tables

Thank You,
Sofiullaha Iqbal Kiron.

Unit 1

Quotes

Examples are better than 1000 words.

Everybody should learn how to program a computer because it teaches how to THINK.

Steve Jobs

Winner **Muhammad Ali**

I hated every minute of training. But I said, "don't quit, suffer now and live the rest of your life as a champion."

Just practice and keep learning...

Juma Ikangaa, marathoner

The will to win means nothing without the will to prepare.

Unit 2

C++, Problems and Algorithm

2.1 Evolution of C

Timeline of language development

Year	C Standard[9]
1972	Birth
1978	K&R C
1989/1990	ANSI C and ISO C
1999	C99
2011	C11
2017/2018	C18

Figure 2.1: Evolution Graph of C

C supports variable sized arrays from C99 standard.

2.2 Array

Here we will discuss about array data structure.
Reference: GfG

2.2.1 Array rotation

2.2.2 Making subarray

A subarray is a contiguous part of an array. An generated array that are already part of an another array.
e.g. : A given array - 1, 2, 3, 4
Subarray of this array: 1, 2, 3, 4, 1, 2, 2, 3, 3, 4, 1, 2, 3, 2, 3, 4, 1, 2, 3, 4
Given arrays are subarray of given array. A array holds $\frac{n(n+1)}{2}$ subarrays where n is the number of elements in main array.

2.3 Memory Allocation

2.3.1 Memory Layout of C

Typical memory layout of C consists with following segments:-

1. Text segment
2. Initialized data segment
3. Uninitialized data segment
4. Stack
5. Heap

For better understanding, look at the block diagram below:-

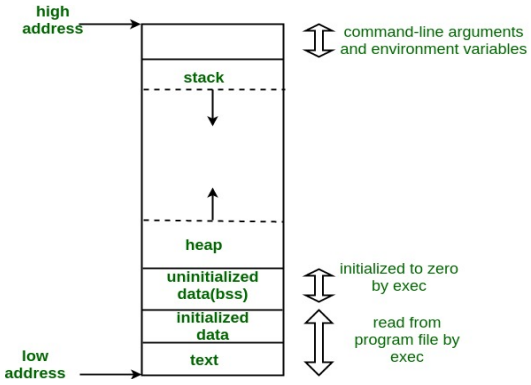


Figure 2.2: C memory layout, GfG

Short explanation:

→ **A text segment**, known as code segment or simply text holder, is one of the section of a program that contains executable instructions as a text. As a memory region, a text segment may be placed below the heap or stack in order to prevent heaps and stack overflows from overwriting it.

→ **Initialized data segment** usually called **Data Segment (DS)**, is a portion of virtual address space of a program, which contains *global variables* and *static variables* that are initialized by the programmer.

→ **Uninitialized data segment**, generally known as **BSS**, named after an ancient assembler operator that stood for "*block started by symbol*". It contains all global variables and static variables that are initialized to zero or do not have explicit initialization in source code.

→ **Stack** is the segment for storing non-static and local variables.

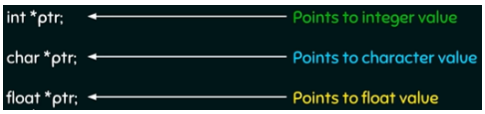
→ **Heap** is the segment where dynamic memory allocation usually takes place. The heap area began at the end of the BSS segment and grows to larger addresses from there.

Wanna learn more?, follow the LINK.

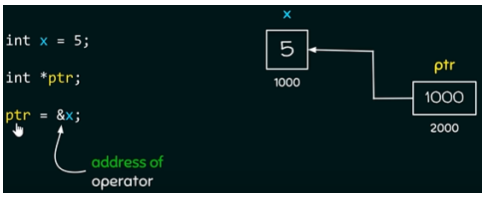
2.3.2 Pointers

NESO Academy
Pointers is a special type of variables which is capable to store initial address of a variable which is points to. That's mean, x is a int variable and that takes byte in memory of the location at 1002 and 1003. Then a pointer that points x will return 1002, the initial point.
General syntax for declaring pointer:-

data_type *pointer_name;



We can assign values to the pointer by address_of operator. Which is Ampersand.



We know that, pointer is also a variable so that it also takes place in memory and then store address of another variable in it.

2.3.3 Dynamic Memory Allocation

Dynamic memory allocation in C/C++ refers to performing allocate memory manually by the programmer. Dynamically allocated memories allocated on **HEAP**.

> Dynamic memory allocation is the process of assigning memory cell during execution time of the program.
>
> > Run-Time memory allocation.

New and Delete operations

Where C uses malloc() function for dynamically allocate memory and free() for remove this. malloc() is a system function which allocates memory on the heap and returns a pointer to the new block. C++ standard supports these functions but also have two extra operator namely new and delete for performing same task in a better and easier way.
New is an operator which denotes a request to memory for allocate space on Heap. **New** operators initializes the memory and returns address of the newly allocated and initialized memory to the pointer variable.

2.4 Illustration of C++ code

If there some problem with the STL of C++, then made own build in functions. Text before ... LaTeX

```
for(int i=0; i<iterations; i++)
{
 do something;
}
```

Text after it ...

2.4.1 Finding maximum element from an array of size n

```
int MaxEle(int arr[], int n)
{
    int i, a=0;
    for(i=0; i<n; i++)
    {
        a = max(a, arr[i]);
    }
    return a;
}
```

2.4.2 Counting selected element from an array

```
int Count(int arr[], int n, int value)
{
    int c=0, i;
    for(i=0; i<n; i++)
    {
        if(arr[i]==value)
        {
            c++;
        }
    }
    return c;
}
```

Table 2.1: Odd numbers in binary

ODD Decimal	in Binary
1	1
3	11
5	101
7	111
9	1001

2.4.3 Divide function by string

```
#include <bits/stdc++.h>
using namespace std;

string divide(string &ns, int &dividor, int &rem)
{
    int i;
    string result;
    rem = ns[0] - '0';
    for (i = 1; i < ns.size(); i++)
    {
        if (rem < dividor)
        {
            rem = rem * 10 + (ns[i] - '0');
            i++;
        }
        result.push_back(rem / dividor + '0');
        rem %= dividor;
    }
    return result;
}

int main()
{
    string ns;
    int dividor, rem;
    rem = 0;
    cin >> ns >> dividor;
    cout << ns << "/" << dividor << " = "
         << divide(ns, dividor, rem) << endl;
    cout << "Reminder: " << rem << endl;
}
```

2.4.4 BitWise Operator

Works at bit-level. Do we know that? Every odd number in binary representation hold true bit in first and last bit. Is it clear? OK. As reverse way, all

Table 2.2: Even numbers in binary

EVEN Decimal	in Binary
2	10
4	100
6	110
8	1000
10	1010

even number in binary representation hold first true bit and false in last. For the sake of illustration:-
Bitwise operators works on binary bits of given number.

Some Interesting things about bitwise operator

1. 'N' is a given number, N will be odd if bitwise operation between N and 1 is 1. Means: (N&1)=1, if N if odd. Else 0 if N is even.

2. The left shift and right shift operators should not be used for negative numbers.

3. We can find odd occurring number in an array by XOR.

```
s;kldf
```

2.4.5 Structure

```
#include<bits/stdc++.h>
using namespace std;

///Now we gonna learning structure in C++.
/*
    Sometimes we need a group of different types
    data in one specific class collection. For get
    released from this problem, C and C++ provides
    a new user-defined data-type.
    To define a structure, use the keyword "struct".
*/
struct student
{
    /*
        All members of struct is generally public.
    */
    int ID;
    string sex;
};

int main()
{
```

```
23      /*
            Let's declare a student struct of class 6 that
25          has 3 students.
            First student(class6[0]) has 33 as roll.
27      */
        student class6[3];
29      class6[0].ID=33;
        class6[1].ID=34;
31      cout << class6[0].ID << endl;
        cout << class6[1].ID << endl;
33      student kiron;
        cin >> kiron.ID;
35      cout << "Kirons id " << kiron.ID << endl;
        cin >> kiron.sex;
37      cout << "Kirons sex " << kiron.sex << endl;

39      /*
            Let's declare a pointer of type student.
41          That can store address of student type variables.
        */
43      student *ptr;
        /*
45          Now This pointer will store address of kiron's all
            member function.
47      */
        ptr = &kiron;
49      /*
            Now ptr points to all member variable of
51          struct variable kiron.
        */
53
        kiron.ID=65;
55      ptr->sex="Male";
        cout << ptr->ID << endl;
57      cout << kiron.sex << endl;
        /*
59          Arrow operator is used for access by pointer.
        */
61  }
```

Listing 2.1: Structure in C/C++

2.4.6 How to check efficiently that a number is palindrome or not

Link

Intuition The first idea that comes to mind is to convert the number into string, and check if the string is a palindrome, but this would require extra non-constant space for creating the string which is not allowed by the problem description.

Second idea would be reverting the number itself, and then compare the number with original number, if they are the same, then the number is a

palindrome. However, if the reversed number is larger than **int.MAX**, we will hit integer overflow problem.

Following the thoughts based on the second idea, to avoid the overflow issue of the reverted number, what if we only revert half of the **int** number? After all, the reverse of the last half of the palindrome should be the same as the first half of the number, if the number is a palindrome.

For example, if the input is 1221, if we can revert the last part of the number "1221" from "21" to "12", and compare it with the first half of the number "12", since 12 is the same as 12, we know that the number is a palindrome.

Algorithm
First of all we should take care of some edge cases. All negative numbers are not palindrome, for example: -123 is not a palindrome since the '-' does not equal to '3'. So we can return false for all negative numbers.

Now let's think about how to revert the last half of the number. For number 1221, if we do 1221 % 10, we get the last digit 1, to get the second to the last digit, we need to remove the last digit from 1221, we could do so by dividing it by 10, 1221 / 10 = 122. Then we can get the last digit again by doing a modulus by 10, 122 % 10 = 2, and if we multiply the last digit by 10 and add the second last digit, 1 * 10 + 2 = 12, it gives us the reverted number we want. Continuing this process would give us the reverted number with more digits.

Now the question is, *how do we know that we've reached the half of the number?*

Since we divided the number by 10, and multiplied the reversed number by 10, when the original number is less than the reversed number, it means we've processed half of the number digits.

2.5 STL Map

Map is a associated container of C++ STL(Standard Template Library). A map variable has two part,

1. Key
2. Value

Geeks for Geeks: Maps are associative container that stores data in a mapped fashion. No two map values can have same key values.

```
#include<map>
// Declaring Map
```

```
map<int, int> mp; // mp: map name.
// First one is keyvalue, second one is mapvalue.

// Insert elements in random order.
mp.insert(pair<int, int>(1, 40));

//Creating map iterator:
map<int, int> :: iterator it;

//Accessing the map data:
it -> first; // Refer keyvalue.
it -> second; // Refer mapvalue.
```

2.5.1 Map member function

1. insert() :- The key must be unique.

```
//Syntax:
 map_name.insert({key, value});

 //e.g.
 map<int, int> mp;
 mp.insert({1, 40});
```

2. count() :-

2.6 STL pair

```
#include<iostream>

/*For pair access, include this header file.
utility is a STL.*/
#include<utility>
using namespace std;

int main()
{
    pair<int, int>p1;
    p1.first = 1;
    p1.second = 2;
    cout << p1.first << " " << p1.second << endl;

    ///Declaring pair array with the size 4.
    pair<int, int>p[4];
    //All pair value assigned to 0 automatically.

    p[0].first=1;
    p[0].second=2;
    cout << p[0].first << " " << p[0].second << endl;

    /*Here we not assigned any value to the pair p[1] and p[2].
    So there will print 0*/
    cout << p[1].first << " " << p[1].second << endl;
    cout << p[2].first << " " << p[2].second << endl;
```

```
    /*
     Member function: make_pair().
     This member function assign values to pair directly.
     Thats mean, this is more sexy.
     Syntax: pair_name = make_pair(value1, value2);
    */
    pair<string, double> ps;
    ps = make_pair("My S.S.C result: ", 4.56);
    cout << ps.first << ps.second << endl;

    /*
     Operating pairs.
     1. (==) comparison.
        It will return 1 is those pairs both values are equal otherwise 0.
        As same for all other comparisons like >=, <=, != etc.
    */
    pair<int, int> pair1, pair2, pair3;
    pair1 = make_pair(1, 4);
    pair2 = make_pair(1, 4);
    pair3 = make_pair(2, 4.1);
    /*No matter what type value we assigned,
    It will type casted automatically by the data type as we declared.
    So it will be same as (2, 4).
    */

    if(pair1==pair2)
    {
        cout << "Pairs are same" << endl;
    }
    cout << (pair1==pair2) << endl;
    if(pair1!=pair3)
    {
        cout << "pair1 and pair3 are not same" << endl;
    }
    else
    {
        cout << "pair1 and pair3 are same" << endl;
    }

    /*
     swap()
     Syntax: pair1.swap(pair2);
     It will swap values of both pair.
     swap pair1.first with pair2.first then
     swap pair1.second with pair2.second
    */
    pair1.swap(pair3);
    cout << pair1.first << " " << pair1.second << endl;

    /*
     swap() isn't working in code::blocks but in other
     online compiler, it works perfectly.
     */
}
```

2.7 STL List

Lists are sequence of elements stored in a linked list. Compared to vectors, they allow fast insertions and deletions, but slower random access.

2.7.1 Member Functions of List

Function	Task
assign	assign elements to list
begin	returns an iterator to the beginning of the list
back	returns a reference to last element of list
clear	removes all elements from the list(Clears the list)
empty	returns true if list is empty
end	returns an iterator just past the last element of a list
erase	remove specific element from list
front	returns a reference to the first element of a list
insert	insert elements into the list
max_size	returns the maximum number of elements that the list can hold
merge	merge two lists
push_back	add an element at end of the list
push_front	add an element at beginning of the list
pop_back	removes the last element of the list
pop_front	remove first element of the list
rbegin	return reverse begin iterator
rend	return reverse end iterator
remove	remove elements from list
remove_if	remove element conditionally
resize	change size of the list
reverse	reverse the list
size	returns the numbers of elements present in the list
sort	sorts the list in ascending order
splice	merge the lists in constant time
swap	swap two list with each other with all elements
unique	removes duplicate element

2.7.2 Discuss

```
vector<int> v(5, 42);
```

It will produce a vector named v with five times 42.

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    vector<int> v(5, 42);
    for(int i=0; i<v.size(); i++)
```

```
    {
        cout << v[i] << " ";
    }
    cout << endl;
}
```

Output: 42 42 42 42 42

2.7.3 assign

assign member function is used assign values from one to another.

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    vector<int> v1;
    for(int i=0; i<10; i++)
    {
        v1.push_back(i);
    }
    vector<int> v2;
    v2.assign(v1.begin(), v1.end());
    for(int i=0; i<v2.size(); i++)
    {
        cout << v2[i] << " ";
    }
    cout << endl;
}
```

2.8 STL Stack

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

/*
 Stacks are a type of container adaptors with LIFO(Last IN First Out).
 So we can say, stack is Stup.
*/

int main()
{
    /*Let's create a stack of int type.*/
    stack<int> s;

    /*Member function push() will add the new element at the top
    of the stack.
    Time complexity O(1)*/
    s.push(10);
    s.push(9); /*9 stored above of 10*/
```

```
    s.push(8);
    s.push(1); /*Now 1 is the top most element.*/

    /*Algorithmic function size() returns current size of
    this stack.*/
    cout << s.size() << endl;

    /*Member function top() will return topper element or that element
    we entered last.
    For this example, it is 1*/
    cout << s.top() << endl;

    /*Member function pop() will delete the topmost element.*/
    s.pop();

    /*After removing topmost element,
    present topmost element is 8*/
    cout << s.top() << endl;

    /*After removing one element,
    current size of stack is 3*/
    cout << s.size() << endl;

    /*Algorithmic function empty() will return true if there is
    no element at the stack.*/
    if(!s.empty())
    {
        cout << "Stack is not empty" << endl;
    }

    return 0;
}
```

Listing 2.2: Stack

```
4
1
8
3
Stack is not
empty
```

Output

2.9 Power of 2

2.9.1 Binary Transform of 2^n

Look at the table:-
(LaTeXprovides a large set of tool for formatting tables)

2^n	Decimal	Binary	Seems like
2^0	1	1	10^0
2^1	2	10	10^1
2^2	4	100	10^2
2^3	8	1000	10^3
2^4	16	10000	10^4
2^5	32	100000	10^5

It's clear that power of 2 in binary holds just one and only one true bit. So, we can simply say that, a number will be power of 2 if its binary holds just one true bit at first.

2.9.2 Algorithm

1. Take a decimal number string "ds".
2. Convert the number into binary string by a function.
   (a) Return binary string name such as "bs".
3. Start checking from the index 1.
4. If find out a true bit

   ```
   return false;
   ```

   That mean, this is not power of 2.
5. Else: Till to end, there is no true bit.

   ```
   return true;
   ```

   That mean, this is power of 2.

2.10 Linked Lists

Reference: Hacker Rank
Today we are going talk about linked lists. It's essentially just a sequence of the element, where each element links to the next elements which next element links to the next elements. A linked list can contain pretty much any kind of data - string, char, int. The elements can be sorted or unsorted, duplicates or unique. If we want to access element from linked list, we need linear time where array give us constant time, because for access array element, just

boom, instantly do that. There are two type of linked list. Singly linked list and doubly linked list. Doubly linked list is a alternate version of singly linked list and access of elements from doubly link list is pretty much easy. One doubly linked list element has two part. One is previous node a second is next element. That mean, each elements having a link to the next element, each elements also linked to the previous element. So, for certain operations, this can be quite handy.
Geeks for Geeks
Advantages over arrays:-

- Dynamic size
- Ease of insertion and deletion

Drawbacks:-

1. Random access not allowed. We have to access elements sequentially from the first node called HEAD. So we can't do binary search with linked lists efficiently with its default implementation.
2. Extra memory allocation required for a pointer with each elements of the list.
3. As array, elements are not stored in contiguous location.

Representation:-
A linked list is represented by a pointer to the first node of the linked list. If the linked list is empty, then the value of the HEAD is NULL. A node consist with two parts:-

- Data
- Reference to the next node, i.e. pointer*.[1]

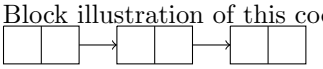
Let's create a user-defined data-type for build a node. A node has two part, one data and another pointer that points to the next node(That also has two part). So, the pointer must be such type that holds the next node(with two part). Means the pointer must be a node object. IN C, node represented by struct. IN object-oriented language, it does with Class. Here the implementation in C++.

```
class Node
{
 public:
        int data;
        Node *next;
};
```

Listing 2.3: Node Class

[1] *A pointer is a special type of variable that stores memory address of another variable. Pointer is declared by asterisk sign before which variable we want to declare that it is a pointer. We can access address of a variable by using ampersand sign before it.

Block illustration of this code:-



2.11 Fortran

Fortran means Formula Translation. This is a general-purpose, compiled imperative programming language that's are specially suited to numeric computation and scientific computing, originally developed by IBM.
(**IBM** - *International Business Machines Corporation.*)

2.12 Python

In Linux and Mac operating systems, python are already installed. For install in windows, first we need to go Pythons Official Website. Then download python and install.

2.12.1 Why we use Python

Easy to learn, easy to use Popular in web development Can create a full web application

Unit 3

Java

Java is an open source high-level, general-purpose(hard & used to develop any kind of programs), object oriented language. If you wanna be a software developer than you must learn that 4 languages:-

1. Python
2. Java
3. JavaScript
4. C / C++

In those, Java is second dominant (and popular also) language that you must learn if you wanna be a platform-independent developer. Also:-

→ Java has huge online community for getting help.

→ Can be used in android development that is most preferable thing today.

Gosling explains his motivation for creating JAVA: James Gosling, father of **JAVA**. It is one of the world's most widely used programming language. It used over ten billion of devices and become central to the development of Android at Google. According to **Oracle**, there are 51 billion active Java Virtual Machines deployed globally.
Java has two types of error. Syntax error and Semantic error. Syntax error is grammatical error and semantic error is that the line has no meaning.

→ "I are playing" - that's the syntax error.

→ "He is hello" - this line has no meaning, semantic error.

3.1 IntelliJ IDEA: The IDE

IntelliJ IDEA is the greatest **IDE** for writing and compiling Java source codes. Here in table 3.1 has some command and shortcuts of **IntelliJ IDEA** that are applicable to windows operating system.

IntelliJ IDEA Commands and Shortcuts		
1	Debug Program	shift + f9
2	Run Program	shift + f10
3	Format Codes	ctrl + alt + L
4	public static void main()	main / psvm
5	System.out.println()	sout
6	To warp a code block in a construct	ctrl + alt + T
7	Search anything in project	double click shift (For a technical reason, I turned off the system)
8	Invoke commit changes dialog	ctrl + K
9	Select several words and edit together	press and hold shift + alt and double click on the word.
10	Fill the code construct	start typing method declaration and press ctrl + shift + enter
11	Join two statement in one line and remove unnecessary spaces	ctrl + shift + J
12	Jump highlighted syntax error	f2 / shift + f2
13	ctrl + shift + E	recently viewed or changed code fragment
14	Change name of the function from main method	highlight method name then press shift + f6 and re-write
15	Quick scheme	view — Quick Switch Scheme or ctrl +
16	To evaluate any expression while debugging program	select the expression in the editor and press Alt + F8

17	Create a new class	alt + ins (Now, not working perfectly)
18	To see all the live template that are valid for this current method or context	ctrl + J (Close this by pressing "esc")
19	Complete syntax with dot from code completion helper(highlighted)	ctrl + dot
20	To close all editor tabs except current one	keep alt pressed and click cross button of current tab
21	To see how to call an action	ctrl + shift + A
22	To select multiple fragment in column mode	press alt + shift + insert and then hold ctrl + shift + alt and drag the mouse
23	Rename an identifier and change all uses of it	Look at the leftmost side(gutter), here is a pencil marker. Click on this and click again Rename.
24	To select a block of code	Double click of its opening brace.
25	Invoke rename action box	mark down the field and press shift + f6
26	Move statement up/down in current block only	put cursor at the end of the statement and hold shift + ctrl then press uparrow/downarrow
27	Move line up/down at any place	shift + alt + up/down
28	To scroll a file horizontally	Turn the mouse wheel while keeping Shift pressed
29	Compare to different directories	select the file on form project (Press and hold shift and then one click on the directory) and press ctrl + D
30	To delete the whole line at the caret	ctrl + Y

Commands and Shortcuts: Intellij IDEA

3.2 Syntax Difference between C++ and Java

3.3 Arrays Class

Arrays class are used to manipulate arrays. Arrays class is a member of the java Collections Framework. We can initialize Array like this, `int anArray[];` which is not conventional way.
Iterate over array by a stream:

```
Arrays.stream(new int[] 1, 2, 3).forEach(n -> System.out.print(n + " "));
```

C++	Java
string	String
int main()	public static void main()
empty()	isEmpty()
vector	ArrayList()

Table 3.2: Syntax Difference between C++ and Java

3.3.1 parallelSort() Method

Java 8 introduced a new method called **parallelSort()** in java.util.Arrays class. If uses parallel sort algorithm. This method uses "MultiThreading" which makes the sorting faster as compared to normal sorting method. This is a void method. Time complexity $O(n \log(n))$

Algorithm for parallelSort():

1. The array is divided into sub-arrays and that sub-array is again divided into their sub-arrays. Until the minimum level of detail in a set of array.
2. Arrays are sorted individually my multiple thread.
3. This algorithm uses Frok / Join concept for sorting.
4. Sorted sub-arrays are then merged.

3.4 Matcher Class

Reference: Oracle Help Center
I have written some descriptions for this class in my Java notebook.

Package java.util.regex

Properties An engine that perform match operations on a character sequence by interpreting patter(Such a regular expression or regex.)

3.5 Interface

Interface is a feature that implements most futuristic class. In java programming language, an interface is a reference type, similar to a class, that can contain only constants, method signatures, default methods, interfaces cannot be instantiated - they can only be implemented by classes or extended by other interfaces. Extension is discussed later in this lesson. All method had just signature but not method body. Method terminated by a semicolon.

3.6 Anonymous Class

Anonymous class enables you to make your code more concise. They enable you to declare and instantiate a class at the same time. They are look like local classes except that they don't have a name. Use them if you need to use a local class only once.

3.7 Lambda Expression

Lambda Expression enables you to pass functionality as an argument to another method. Such as, what action should be taken when someone clicks a button.

Note of Collection: A Collection is an object that groups multiple elements into a single unit by a wrapper class for primitives and instant class for local class objects. Collections are used to store, retrieve, manipulate and communicate aggregate data. As, a List is an Collection of ordered type.

3.8 Object Ordering

```
Collections.sort(list)
```

Starting: There is an interface named *Comparable* and has a *compare* method at this. we can use(override) this method for our own manipulation of sorting / ordering. Comparable implementations provide a *natural ordering* for a class, which allows objects of that class to be sorted automatically. If you try to sort a list, which don not implement Comparable, will throw a **ClassCastException**. Similarly, *Collections.sort(list)* will throw the same exception if we try to sort that list whose elements cannot be compared to one another.

> Elements that can be compared to one another are called *mutually comparable.*
>
> - Oracle

3.8.1 Implement own comparable type

The comparable interface consist of the following methods:

```
public interface Comparable<T>
{
        public int compareTo(T o);
}
```

This method compares the receiving object with the specified object and returns negative, zero or positive value depending on whether the receiving object is less than, equal or greater than the specified object.

```
public class Name implements Comparable<Name>
{
    private final String firstName, lastName;

    public Name(String firstName, String lastName)
    {
        if (firstName == null || lastName == null)
            throw new NullPointerException();

        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String firstName() {return firstName;}
    public String lastName() {return lastName;}

    public boolean equals(Object o)
    {
        if (!(o instanceof Name))
            return false;

        Name n = (Name) o;
        return n.firstName.equals(firstName) && n.lastName.equals(lastName);
    }

    public int hashCode()
    {
        return 31 * firstName.hashCode() + lastName.hashCode();
    }
    public String toString() {return firstName + " " + lastName;}

    public int compareTo(Name n)
    {
        int lastCmp = lastName.compareTo(n.lastName);
        return (lastCmp != 0 ? lastCmp : firstName.compareTo(n.firstName));
    }
}
```

3.8.2 Comparators

What if you want to sort some objects in an order other than their natural ordering? Or what if you want to sort some objects that don't implement Comparable? To do either of these things, you'll need to provide a Comparator - an object that encapsulates an ordering. The Comparator interface consists of a single method:

```
public interface Comparator<T>
{
        int compare(T o1, T o2);
}
```

The *compare* method returns less than zero, zero or greater than zero depending on whether the first argument is less than, equal or greater than the second.

By GeeksforGeeks: A *comparator* is an interface[1] that used to order the objects of user- defined classes. A *comparator* object is capable of comparing two objects of two different classes. Suppose we have an "ArrayList" of objects containing fields like NAME, ROLL, SECTION and we need to sort the "ArrayList" based on ROLL. We can do this by:

Method 1: One obvious approach is to write our own "sort()" function using one of the standard algorithm. This approach require rewrite the entire function.

Method 2: Using *comparator* interface[2]. It contains two method signature:-

(a) compare(Object o1, Object o2);

(b) equals(Object element);

How does "Collections.sort()" work? Internally the "sort()" method calls Compare method of that class and asks for "which is grater?". Then compare method returns $-1, 0$ or 1 based on compared object. It uses this result to then determine if they should be swapped for its sort.

A working program: A working program from geeksforgeeks

```
class Student
{
    String name, address;
    int roll;

    public Student(String name, int roll, String address)
    {
        this.name = name;
        this.roll = roll;
        this.address = address;
    }

    public String toString()
    {
        return this.name + " " + this.roll + " " + this.address;
    }
}

class SortByName implements Comparator<Student>
{
    public int compare(Student s1, Student s2)
    {
        return s1.name.compareTo(s2.name);
```

[1] That just had a face, no body.

[2] Present in java.util package.

```
    }
}

class SortByAddress implements Comparator<Student>
{
    public int compare(Student s1, Student s2)
    {
        return s1.address.compareTo(s2.address);
    }
}

class Main
{
    public static void main(String[] args)
    {
        ArrayList<Student> list = new ArrayList<>();
        list.add(new Student("Kiron", 65, "Madaripur"));
        list.add(new Student("Miron", 73, "Faridpur"));
        list.add(new Student("Chiron", 11, "Dhaka"));

        Collections.sort(list, new SortByName());
        for (int i = 0; i < list.size(); i++)
            System.out.println(list.get(i));

        System.out.println();
        Collections.sort(list, new SortByAddress());
        for (int i = 0; i < list.size(); i++)
            System.out.println(list.get(i));
    }
}
```

Listing 3.1: Comparator for sorting Student class

Output

```
Chiron 11 Dhaka
Kiron 65 Madaripur
Miron 73 Foridpur

Chiron 11 Dhaka
Miron 73 Foridpur
Kiron 65 Madaripur
```

Just change the return value inside compare method, that will allow us to sort objects in required manner. e.g. for descending order, swap the position of objects.

Again Oracle:

3.9 Math Class

3.9.1 Beyond Math Class

The "Math" class in *java.lang* package provides methods and constants for doing more advance mathematical computation. All the methods in "Math" class are "static", so they can be called directly. Using the **static import** language feature, you don't have to write Math in front of every math function.

```
import static java.lang.Math.*;
```

* will import all the methods from the "Math" class.
Now we can write code like this:

3.9.2 Constants and Basic Methods

The class contains two constants:

Math.E Base of natural logarithms.

Math.PI Ratio of the circumference of a circle to its diameter.

This class also includes more than 40 static methods.

3.9.3 Random Numbers

The random() method returns a pseudo-randomly selected number between 0.0 and 1.0

primitive abs(primitive arg) Returns the absolute value of "arg".

ceil()

floor()

double rint(double d) Returns the integer that is closest in value to the argument as double.

1. min(a, b);
2. max(a, b);
3. double exp(double d);
4. double log(double d);
5. pow(base, exponent);
6. sqrt(double d);
7. All trigonometric methods.

double atan2(double y, double x) Converts rectangular coordinates (x, y) to polar coordinate (r, θ) and returns θ.

8. double toDegrees(double d);
9. double toRadians(double d);

con(angle);

3.10 Graph

3.10.1 Representation

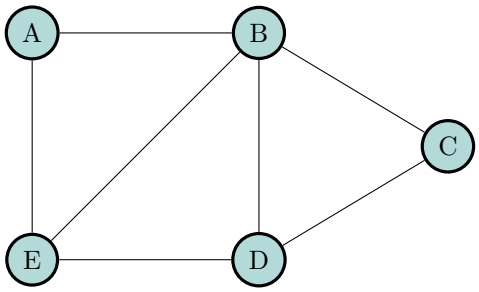
What is graph: Graph is a collection of nodes / vertices(**V**) and edges(**E**) placed between them. Represented as:

$$G = (V, E)$$

We can traverse vertices through edges. These edges might be weighted or non-weighted.

There are two kinds of graphs:

Un-directed: Not direction specified. We can traverse either direction between two vertices.



Directed: Can traverse only in the specified direction between that two vertices.

Two common ways to represent a graph:

- Adjacency Matrix
- Adjacency List

Adjacency Matrix: Adjacency matrix is a 2D array which had the size of $(V * V)$, where V are the number of vertices in the graph. Adjacency matrix is symmetric if the graph is un-directed. Here is the adjacency matrix for above graph:

	A	B	C	D	E
A	0	1	0	0	1
B	1	0	1	1	1
C	0	1	0	1	0
D	0	1	1	0	1
E	1	1	0	1	0

3.11 Find Integer Root

I got a problem from *Leetcode* online judge. That wanna from us an integer root. Here is some attempt to reach the solution.

Natural Number In mathematics, the natural numbers are those used for counting[3] and ordering[4]. Normally natural numbers starts from 0 and run as:- $1, 2, 3, 4, \ldots$

Integer square root In number theory, the *integer square root* $isqrt(n)$ is a positive integer m where $m <= n$. Or,

$$isqrt(n) = \lfloor \sqrt{n} \rfloor$$

[3]There are *six* coins on the table: Cardinal Numbers
[4]This is the *third* largest city in the world: Ordinal Numbers

Unit 4

JavaScript

JavaScript is the programming language of HTML and web :w3schools.
User rating of JS is 67.8%. See? how many people use JS. For become a great software developer or web developer, you must learn **JavaScript**. It seems impossible to be a developer without using JS. JavaScript provides an easy way to create interactive web pages smoothly.
All the functionality of JS in html written between <script>. . . < /script> tag.

4.1 Why We Should Learn JS

JavaScript is one of the 3 languages all web developer must learn.

1. HTML
2. CSS
3. JavaScript(JS)

Web pages are not the only place where **JS** is used. Many desktop and server programs use **JS**.

4.2 Variable

For declaring variable, use **var** keyword before variable name.
Syntax: var variable_name = value;
If value is a string, write string value in single or double quotes. It's variable naming is same as C or C++. JavaScript is case sensitive. That's mean, var lastname and Lastname is not same.

4.3 String

JavaScript accepts both double and single quotes for assigning string values to variable.
var string;
string = "Kiron";
String operation as:
"Iqbal" + " " + "Kiron" is same as "Iqbal Kiron".

4.4 Comments

JS comment is fully same as C and C++ comments. JS supports both One-line and Multiline comment.
Oneline comment: // or ///This is a comment.
Multiline comment: /*...*/

Unit 5

Julia

Julia is a high-level, high-performance, dynamic programming language. Many of its feature are well-suited for numerical analysis and computational science. This language first appeared on 2012: 8 years ago and we get a stable release 1.5.2 on 24 September, 2020. Mainly implemented by C / C++, Scheme and motivated from some high performed languages. Runs on Windows, Linux, macOS and FreeBSD. Licensed form MIT. In this language, all library functions of C and FORTRAN can be call directly. It is a general purpose programming language, originally designed for numerical / technical computing. Indentation and code formatting are closer to Python. Here is a code for implement Sir Isaac Newtons root finding method:

```
x0 = 1          # The initial guess

f(x) = x^2 - 2          # The function whose root we are trying to find

fprime(x) = 2x          # Derivative of f(x)

tolerance = 1e - 7              # 7 digit accuracy is desired

epsilon = 1e - 14               # Do not divide by a number smaller than this

maxIterations = 20

solutionFound = false

for i =1 : maxIterations
        y = f(x0)
        yprime = fprime(x0)

        if abs(yprime) < epsilon
                break
        end

        global x1 = x0 - y / yprime             # Do Sir Newtons computation

        if abs(x1 - x0) <= tolerance
                global solutionFound = true
```

```
            break
```

Unit 6

Coursera Algorithm

Algorithm: Method for solving a problem.

Data Structure: Method to store information.

Algorithms of all of this course are implemented in Java.

Unit 7

Windows Command Prompt

Widows command prompt or windows terminal is a console for creating command line. If you wanna be a great developer then you must get used to uses of this.

Task	Command Code
For clear total screen	cls
Will show all color code	color/?
For start a soft program	start soft_name
Wanna exit from current folder?	exit
Open a file from current folder	cd folder_name
View name of all files in current folder	dir
View name of all files in current folder with hidden files	dir/a
Make a new folder	mkdir folder_name
Back from current folder	cd ..
Back more than one	cd ../.. as this
Remove directory or folder(If the folder is currently empty)	rmdir folder_name
Remove directory or folder(If not empty)	rmdir /s folder_name and then y or YES confirmation.
Compile java source code	javac file_name.java
Run java class	java file_name
Type few first words of file and then automatically filled full name	Press tab key :-not command
Delete all file as same type	del *.extension

Unit 8

Type Setting

8.1 Formatting

Sometimes we need as special symbols that are not available in keyboard. Then go here, click what symbol you need and copy the command from text editor of Tutorials Point.

8.2 Font

Here the picture guide of Font-Size That we can use in "basicstyle" parameter

Font sizes

Command	Output
\tiny	Lorem ipsum
\scriptsize	Lorem ipsum
\footnotesize	Lorem ipsum
\small	Lorem ipsum
\normalsize	Lorem ipsum
\large	Lorem ipsum
\Large	Lorem ipsum
\LARGE	Lorem ipsum
\huge	Lorem ipsum
\Huge	Lorem ipsum

at "lstset".

8.3 Code Listing parameters

Code listing parameters:

language=Octave -> choose the language of the code
basicstyle=\footnotesize -> the size of the fonts used for the code
numbers=left -> where to put the line-numbers
numberstyle=\footnotesize -> size of the fonts used for the line-numbers
stepnumber=2 -> the step between two line-numbers.
numbersep=5pt -> how far the line-numbers are from the code
backgroundcolor=\color{white} -> sets background color (needs package)
showspaces=false -> show spaces adding particular underscores
showstringspaces=false -> underline spaces within strings
showtabs=false -> show tabs within strings through particular underscores
frame=single -> adds a frame around the code
tabsize=2 -> sets default tab-size to 2 spaces
captionpos=b -> sets the caption-position to bottom
breaklines=true -> sets automatic line breaking
breakatwhitespace=false -> automatic breaks happen at whitespace
morecomment=[l]{//} -> displays comments in italics (language dependent)

If you are using several parameters, they have to be separated by commas.

8.4 Shadow frame in code listing

Picture guide of shadow fraem in listing:

The following will draw a frame around your source code with a blue shadow (you will need the color-package).

```
\lstset{frame=shadowbox, rulesepcolor=\color{blue}}
```

If you want closed frames on each page, use the following command sequence:

```
\begin{framed}
\begin{lstlisting}...\end{lstlisting}
or \lstinputlisting{...}
\end{framed}
```

8.5 Themes we can use in BEAMER class

8.5.1 Presentation themes without navigation bar

1. default
2. boxes
3. Bergen
4. Boadilla
5. Madrid
6. AnnArbor
7. CambridgeUS
8. EastLansing
9. Pittsburgh
10. height=1 cm, Rochester

8.5.2 Presentation theme with a tree-like navigation bar

1. Antibes
2. JuanLesPins

3. Montpellier

4. Berkeley

5. Goettingen

6. Marburg

7. Hannover

8. Berlin

9. Ilmenau

10. Dresden

11. Darmstadt

12. Frankfurt

13. Singapore

14. Szeged

8.6 List of Greek letters and math symbols

Greek letters

αA	\alpha A	νN	\nu N
βB	\beta B	$\xi \Xi$	\xi\Xi
$\gamma \Gamma$	\gamma \Gamma	oO	o O
$\delta \Delta$	\delta \Delta	$\pi \Pi$	\pi \Pi
$\epsilon \varepsilon E$	\epsilon \varepsilon E	$\rho \varrho P$	\rho\varrho P
ζZ	\zeta Z	$\sigma \Sigma$	\sigma \Sigma
ηH	\eta H	τT	\tau T
$\theta \vartheta \Theta$	\theta \vartheta \Theta	$\upsilon \Upsilon$	\upsilon \Upsilon
ιI	\iota I	$\phi \varphi \Phi$	\phi \varphi \Phi
κK	\kappa K	χX	\chi X
$\lambda \Lambda$	\lambda \Lambda	$\psi \Psi$	\psi \Psi
μM	\mu M	$\omega \Omega$	\omega \Omega

Arrows

$\leftarrow$	\leftarrow	$\Leftarrow$	\Leftarrow
$\rightarrow$	\rightarrow	$\Rightarrow$	\Rightarrow
$\leftrightarrow$	\leftrightarrow	$\rightleftharpoons$	\rightleftharpoons
$\uparrow$	\uparrow	$\downarrow$	\downarrow
$\Uparrow$	\Uparrow	$\Downarrow$	\Downarrow
$\Leftrightarrow$	\Leftrightarrow	$\updownarrow$	\updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto
$\nearrow$	\nearrow	$\searrow$	\searrow
$\swarrow$	\swarrow	$\nwarrow$	\nwarrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown

Miscellaneous symbols

∞	\infty	$\forall$	\forall
$\Re$	\Re	$\Im$	\Im
∇	\nabla	$\exists$	\exists
∂	\partial	$\nexists$	\nexists
$\emptyset$	\emptyset	$\varnothing$	\varnothing
$\wp$	\wp	$\complement$	\complement
$\neg$	\neg	$\cdots$	\cdots
$\square$	\square	$\surd$	\surd
$\blacksquare$	\blacksquare	$\triangle$	\triangle

Binary Operation/Relation Symbols

$\times$	\times	$\times$	\times
$\div$	\div	$\cap$	\cap
$\cup$	\cup	$\neq$	\neq
$\leq$	\leq	$\geq$	\geq
$\in$	\in	$\perp$	\perp
$\notin$	\notin	$\subset$	\subset
$\simeq$	\simeq	$\approx$	\approx
$\wedge$	\wedge	$\vee$	\vee
$\oplus$	\oplus	$\otimes$	\otimes
$\Box$	\Box	$\boxtimes$	\boxtimes
$\equiv$	\equiv	$\cong$	\cong

8.7 Available Document Structure Commands

8.7.1 BOOK and REPORT

- part
- chapter
- section

- subsection
- subsubsection
- paragraph
- subparagraph

8.7.2 ARTICLE

- part
- section
- subsection
- subsubsection
- paragraph
- subparagraph

8.8 XeLaTeX

XeLaTeX is replacement for PDFLaTeX, it will takes input form LaTeX and turns output into a PDF. You cannot see instant created PDF file besides of TeXMaker.
The major differences between LaTeX/PDFLaTeX are:

- XeLaTeX assumes input is in UTF-8(which allows encoding of non-English and non-Latin characters) by default.
- XeLaTeX uses system fonts, not specially installed LaTeX fonts.

It's packaged with TeXLive distribution of LaTeX.

8.9 The "hyperref" Package

Include as `\usepackage[hidelinks]hyperref` For adding hyperlink / clickable link. *hidelinks* will remove red frame form all type of links. While importing this package, it has to be the last package that are included. Because of, The *hyperref* documentation says, "Make sure it comes last of your loaded packages". The reason is that it redefines many LaTeX commands. It's a rule of thumb that helps you to avoid error. Also that, the *amsrefs* user guide notes that, this package has to become after *hyperref*.


OVER