Programming Document

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

BSMRSTU, Department of CSE
SHIICT

11 April, 2020
5:13pm
Version: 0.0.132

A-Z Menu

List of Figures

List of Tables

Thank You,
Sofiullaha Iqbal Kiron.

Chapter 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.”

You lost?

Just practice and keep learning...

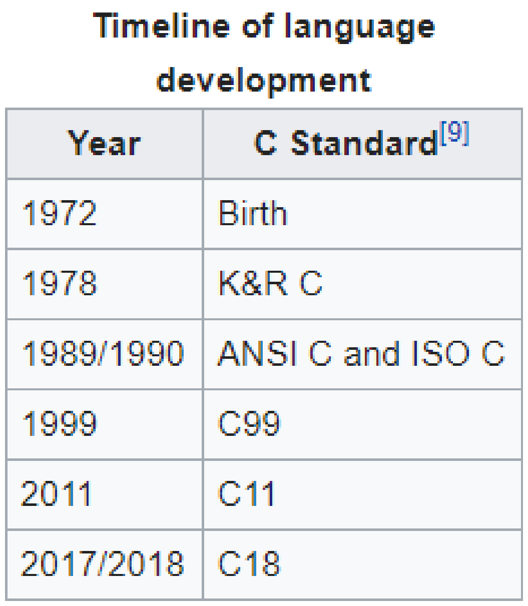
Juma Ikangaa, marathoner

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

Chapter 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:-

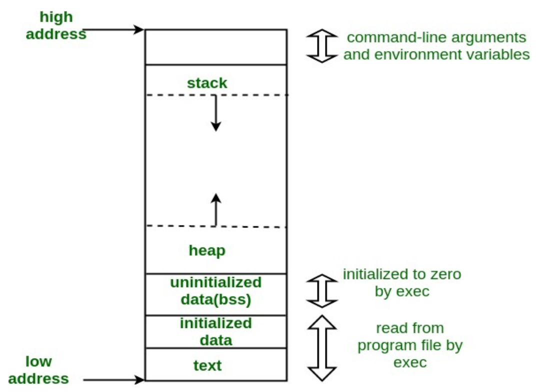

high address
stack
heap
uninitialized data(bss)
initialized data
text
low address
command-line arguments and environment variables
initialized to zero by exec
read from program file by exec

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.

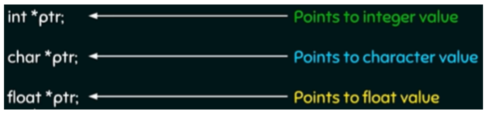
→ **Stack** is the segment for storing non-static and local variables.

→ **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.

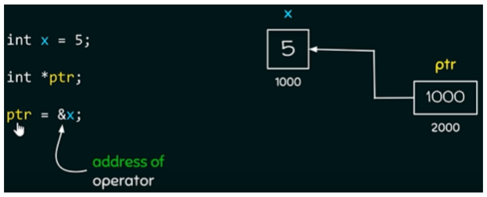
Wanna learn more?, follow the LINK.

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;


```
int *ptr;    <------------- Points to integer value
char *ptr;   <------------- Points to character value
float *ptr;  <------------- Points to float value
```

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


```
int x = 5;

int *ptr;

ptr = &x;
```

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;
}
```

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 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.

Table 2.1: Odd numbers in binary

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

Table 2.2: Even numbers in binary

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

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

    /*
        Let's declare a pointer of type student.
        That can store address of student type variables.
    */
    student *ptr;
    /*
        Now This pointer will store address of kiron's all
        member function.
    */
    ptr = &kiron;
    /*
        Now ptr points to all member variable of
        struct variable kiron.
    */

    kiron.ID=65;
    ptr->sex="Male";
    cout << ptr->ID << endl;
    cout << kiron.sex << endl;
    /*
        Arrow operator is used for access by pointer.
    */
}
```

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:-

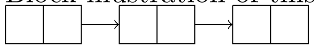
- Data
- Reference to the next node, i.e. pointer*.[1]

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

Block illustration of this code:-



[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.

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

Chapter 3

Java

Java is an open source high-level, general-purpose(Easy & 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:-

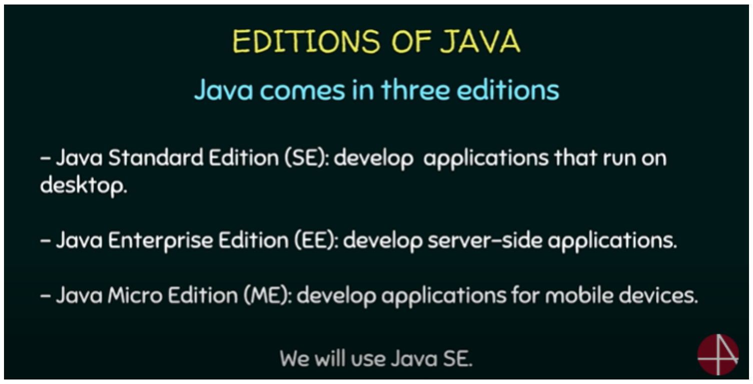
→ Java has huge online community for getting help.

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

Java has two types of error. Syntax and Semantic error. Syntax error is grammar error and semantic error is that the line has no meaning.

→ "I are playing" - this is syntax error.

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



EDITIONS OF JAVA

Java comes in three editions

- Java Standard Edition (SE): develop applications that run on desktop.
- Java Enterprise Edition (EE): develop server-side applications.
- Java Micro Edition (ME): develop applications for mobile devices.

We will use Java SE.

3.1 IntelliJ IDEA

IntelliJ IDEA is a greatest IDE for writing and compiling Java source codes. Here is some command and shortcuts of IntelliJ IDEA for windows operating system.

	IntelliJ IDEA
Debug Program	
Run Program	
Format Codes	
public static void main()	
System.out.println()	
To warp a code block in a construct	
Search anything in project	
Invoke commit changes dialog	
Select several words and edit together	p
Fill the code construct	start
Join two statement in one line and remove unnecessary spaces	
Jump highlighted syntax error	
ctrl + shift + E	
Change name of the function from main method	hi
Quick scheme	
To evaluate any expression while debugging program	
Create a new class	
To see all the live template that are valid for this current method or context	
Complete syntax with dot from code completion helper(highlighted)	
To close all editor tabs except current one	
To see how to call an action	
To select multiple fragment in column mode	press alt + s
Rename an identifier and change all uses of it	Look at the leftmost
To select a block of code	
Invoke rename action box	
Move statement up/down in current block only	put cursor at the e
Move line up/down at any place	
To scroll a file horizontally	
Compare to different directories	select the file on form proj
To delete the whole line at the caret	

3.2 Syntax Difference between C++ and Java

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

Table 3.1: Syntax Difference between C++ and Java

3.3 Matcher Class

Reference: Oracle Help Center

3.3.1 Package

java.util.regex

3.3.2 Properties

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

Chapter 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 at <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 Oneline and Multiline comment.
Oneline comment: // or ///This is a comment.
Multiline comment: /*...*/

Chapter 5

Coursera Algorithm

Algorithm:	Method for solving a problem.
Data Structure:	Method to store information.

Algorithms of all of this course are implemented in Java.

Chapter 6

Windows Command Prompt

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

Chapter 7

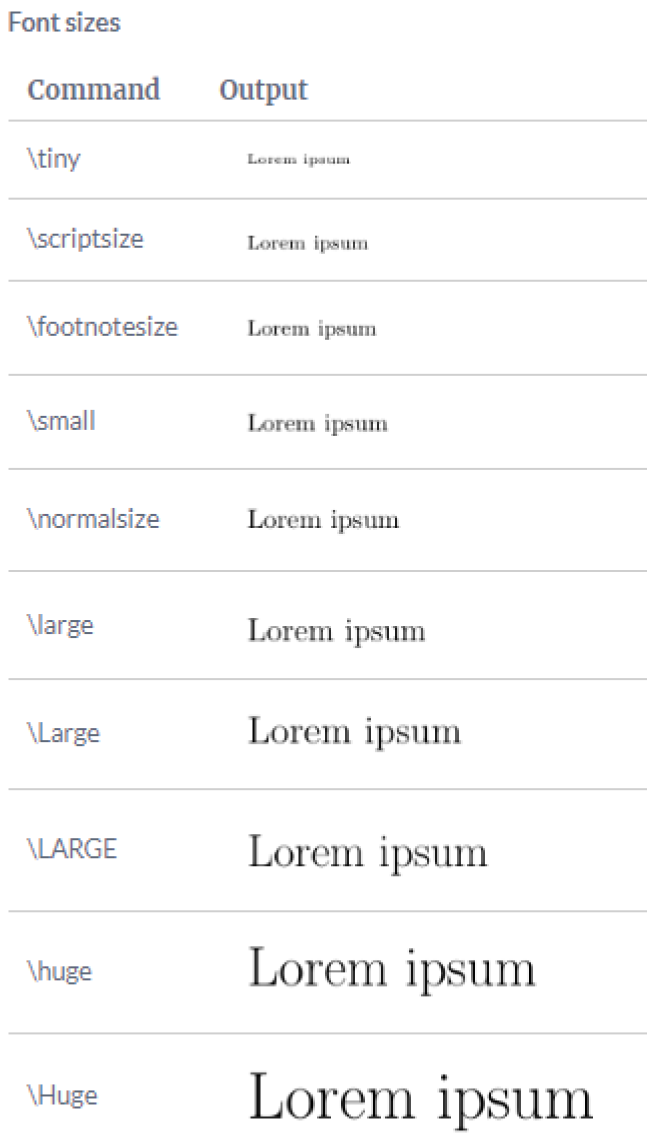
Type Setting

7.1 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.

7.2 Font

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



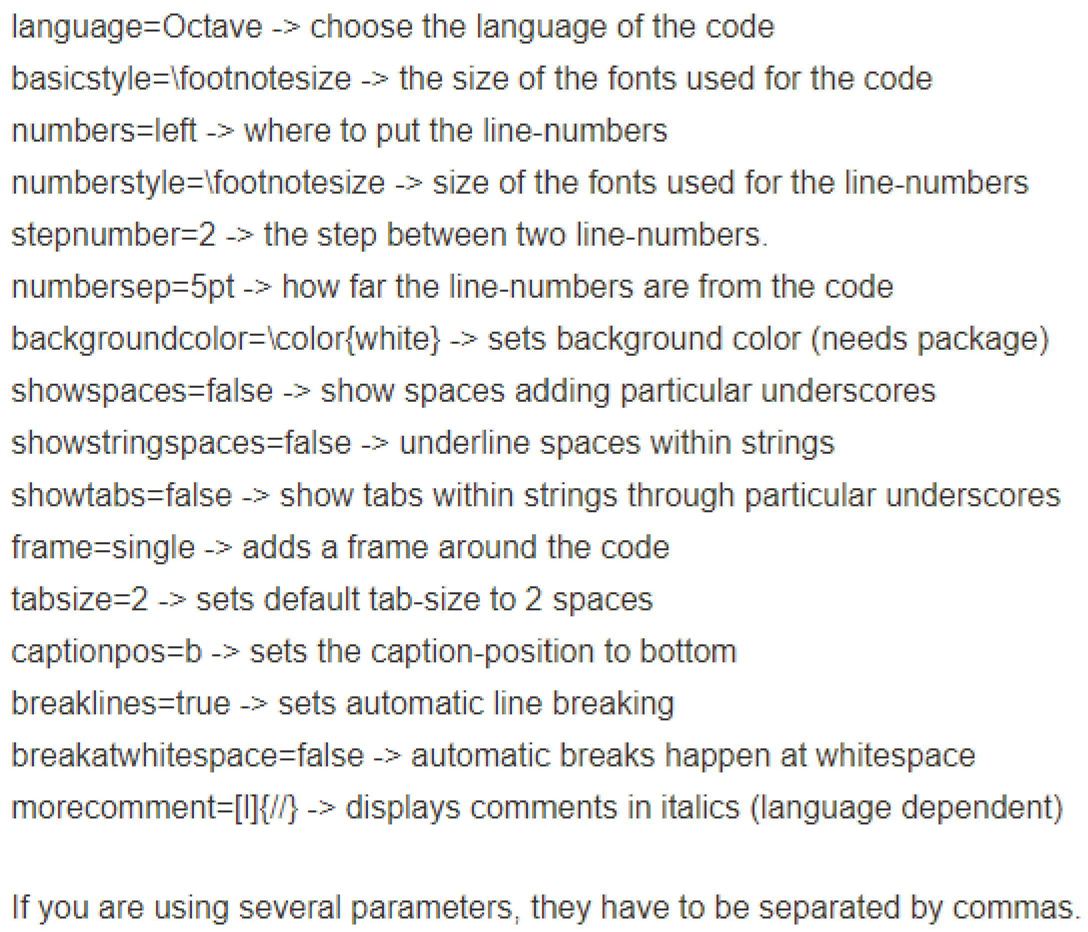
Font sizes

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

rameter at "lstset".

7.3 Code Listing parameters

Code listing parameters:

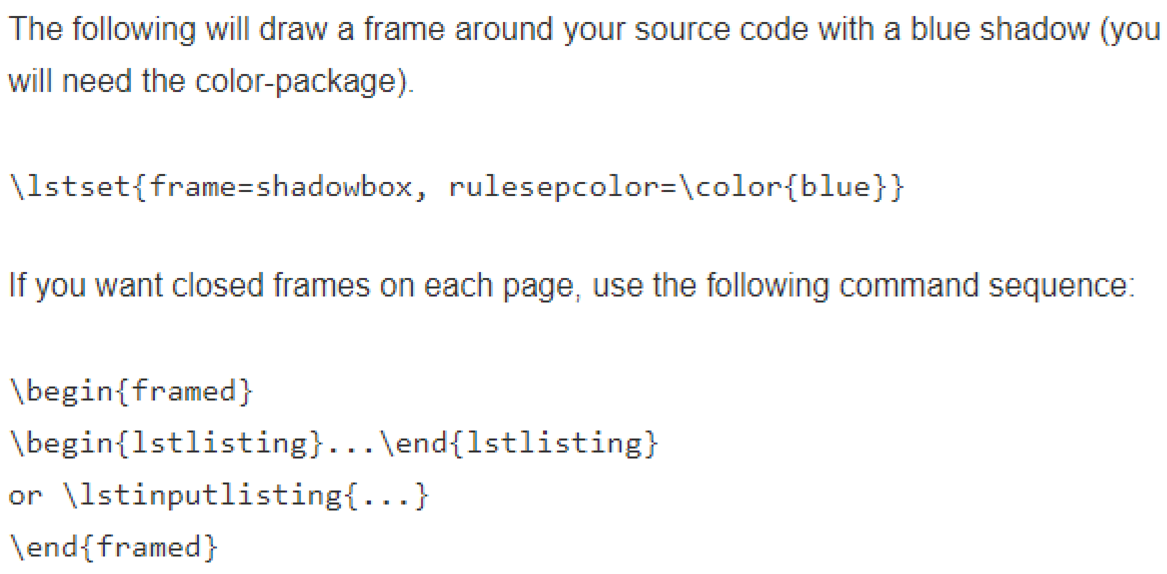


language=Octave -> choose the language of the code
basicstyle=\footnotesize -> the size of the fonts used for the code
numbers=left -> where to put the line-numbers
numberstyle=\footnotesize -> size of the fonts used for the line-numbers
stepnumber=2 -> the step between two line-numbers.
numbersep=5pt -> 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.

7.4 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}
```

7.5 Themes we can use in BEAMER class

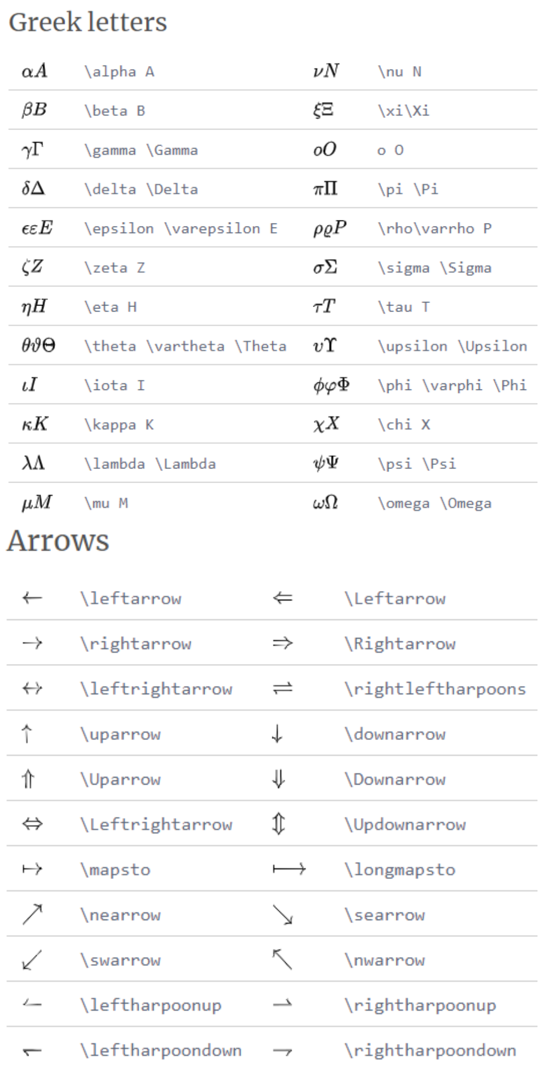
7.5.1 Presentation themes without navigation bar

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

7.5.2 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

7.6 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	$o O$	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`