

Table of Contents

Acknowledgement.....	5
Project Description	6
1. Abstract:	7
2. The Purpose of the Project	8
a. Goals of the Project	8
b. Background of Project.....	8
3. The Scope of the Work	9
a. The Current Situation	9
b. Competing Products	9
4. Stakeholders	10
a. Customer	10
b. Hands-On Users of the Product	10
Requirement	11
1. Product use Cases.....	12
a. Use case Diagram	
b. Product use case List	
2. Functional Requirements	13
3. Non-Functional Requirement.....	14

a. System Requirements.....	14
b. Performance Requirements	14
c. Look and Feel Requirements.....	15

Design.....16

1. Design goals	17
2. Library Diagram	18
3. Object Design	19
4. Library/Software mapping	20
5. User Interface	21

Development.....24

1. Core of DSALGO Library.....	25
2. File Structure.....	26
a. Array's files.....	28
b. Linked List's files.....	32
c. Stack's files.....	34
d. Queue files.....	36
e. Tree's Files.....	38
f. Graph's Files.....	40

g. dsalolib.h file.....	41
-------------------------	----

Testing.....42

1. Unit Testing	43
2. Integration Testing	43
3. System testing	44

Implementation.....45

1. Library Setup	46
2. Documentation of Library	50

References/Bibliography.....	51
------------------------------	----

ACKNOWLEDGEMENT

We take this occasion to thank God, almighty for blessing us with his grace and taking our endeavor to a successful culmination. We extend our sincere and heartfelt thanks to us esteemed guide, RAJESH JHA, for providing us with the right guidance and advice at the crucial junctures and for showing me the right way. We would like to thank our friends and family for the support and encouragement they have given us during the course of our work.

Section:-1

Project Overview

Abstract

Once programmers have grasped the basics of object-oriented programming and C++, the most important tool that they have at their disposal is the data structure and algorithms which gives the accessibility and ease to solve the real world problem.

This can be provided to them with a library of reusable objects and standard data structures. This report shows the complete process of the data structure & algorithm library DSALGO Library provides a carefully integrated use of general data structures and their implementation using easy class, object and their respective function. In doing so, the developer will be able to get to know the functionality of each namespace, class, function and template using the documentation of the library which is well written on the website. Website for downloading the library files and user guidance is developed using one of the trending frameworks Django(python).

Purpose of the Project

In the world of computers, people think the data structure and algorithm are hard to understand and implement. So the objective of our project development is to provide our users easy access to the DS algo library, as well as provide an easy way to implement their concept. Our users (developer or students) should not face any trouble while working on their project and understanding it.

(a) Goal of the Project

The project aims and objectives that will be achieved after completion of this project are discussed in this subchapter. The aims and objectives are as follows:

- Reduce the complexity of the concept
- Reduce the complexity of algorithms
- Easy accessibility
- Website for easy understanding of the developed library
- To accomplish college project

(b) Background of Project

We are developing a library to help programmers and students have the accessibility and ease to solve the problems during programming. The library is developed using c++ and its core concepts such as templating, namespace.

The Scope of the Work

Open Source and Competitive programming is nowadays becoming the craze among the newbie developers.

Every Developer wants to reduce time complexity and to get their work done in no time. Their Data Structure & Algorithm library will be a helpful integration for developers for ease of access.

(a) Current Situation :-

If a developer is choosing the C++ language the main reason for the choice is the speed of C plus plus. Handling data structure and managing them according to project becomes the overhead for developers. Although there is Standard Template Library also but when it comes to time complexity of any algorithm or operation then we have only limited choice there.

Example: - We can't perform heap sort according to our data. So we need more functionality and access to operate over the data.

(b) Competing Product:-

There are rarely any software/libraries that provide concept as data structure & algorithm functionality as wastely. The Standard template library of C++ provides the functionality of the data structure & algorithm but from the point of view of a beginner that can be a headache to understand and use it as per need.

Stakeholder

(a) Targeted Audience :-

Our target audience are mainly those who are using C++ language for their learning purpose or small project implementation.

We can target the two main community in society:-

1. Student related to programming Field
2. New Developer that is implementing their small projects in real life.

(b) Hands on User of the Product/Library:-

Currently we have not that big user base/hands on user but our group who is developing this library and a few of our colleagues and friends that use it on their small projects. There are a few junior students who are currently learning the C++ as an intermediate. They also used it in some of the cases While they were implementing the basic concept of data structure.

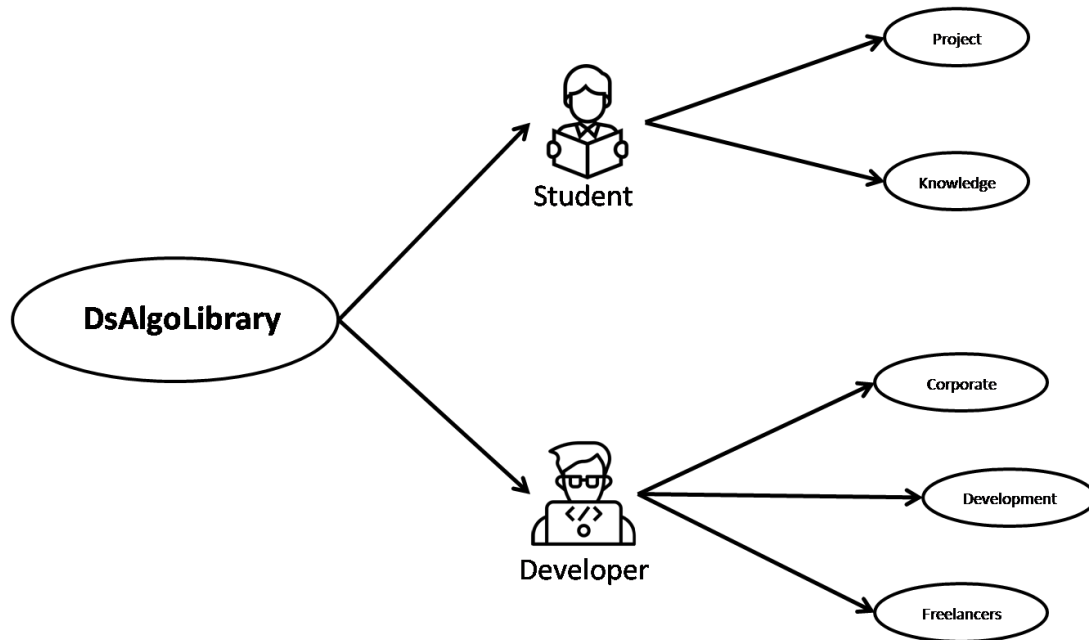
Section-2

Requirement

Products Use Cases

The community of students and developers would be using our DS Library. Students can use it for acquiring some knowledge and for their projects, developers can use it for developing their projects at the workplace.

(a) Use Case Diagram



(a) Produce Use Case List

DsAlgoLibrary

1. Student
 - Knowledge
 - Project
2. Developer
 - Corporate
 - Freelancer
 - Development

Functional Requirements

Functional requirement gives a clear view of functionality of C++ Data Structure & Algorithm Library.

Each type of Data Structure will have the following operation to perform on data of (Integer, char, double,String) type.

1. **Insertion** :- Each data structure should have the functionality of inserting data into the data structure memory.
2. **Deletion**:- Every data structure will have the functionality of deleting specific elements as per the requirement.
3. **Traversing** :- In the Library every data structure should have the functionality to traverse or visit each element
4. **Searching**:- In the library every data structure(tree, linkedlist, stack , etc..) should have the functionality of searching a particular element.
5. **Sorting**:- Each Data Structure will have the functionality of sorting/arranging the data in the particular given format.

Non-Functional Requirements

(a). System Requirements-

Processor	Intel core processor, Amd ryzen ,pentium.
Operating System	Windows-7.1, windows-8,mac-os,ubuntu.
Memory	1GB RAM or More.
Hard Disk	3 GB ROM For running the software on which library will be used.
Software	Code Block.

(b) Performance Requirement:-

- Responses to view information shall take no longer than 5 seconds to appear on the screen.
- Library should be able to store and operate over N^6 number of data, where N = Number of elements in any data structure.

(c) Look and Feel Requirements-

- Library Documentation should have a well developed design that would be user oriented.
- The User Interface of the documentation should be attractive.
- Navigation between multiple pages should be easy.
- Documentation will be written well and understandable.

Section-3

Design

Design Goals

Design goal of the data structure & algorithm library are as follow:-

1. **Correctness:**

A good design should be correct i.e. it should correctly implement all the functionalities of the system.

2. **Efficiency:**

A good library design should address the resources, time, and space optimization issues.

3. **Understandability:**

A good design should be easily understandable, for which it should be modular and all the modules are arranged in layers.

4. **Completeness:**

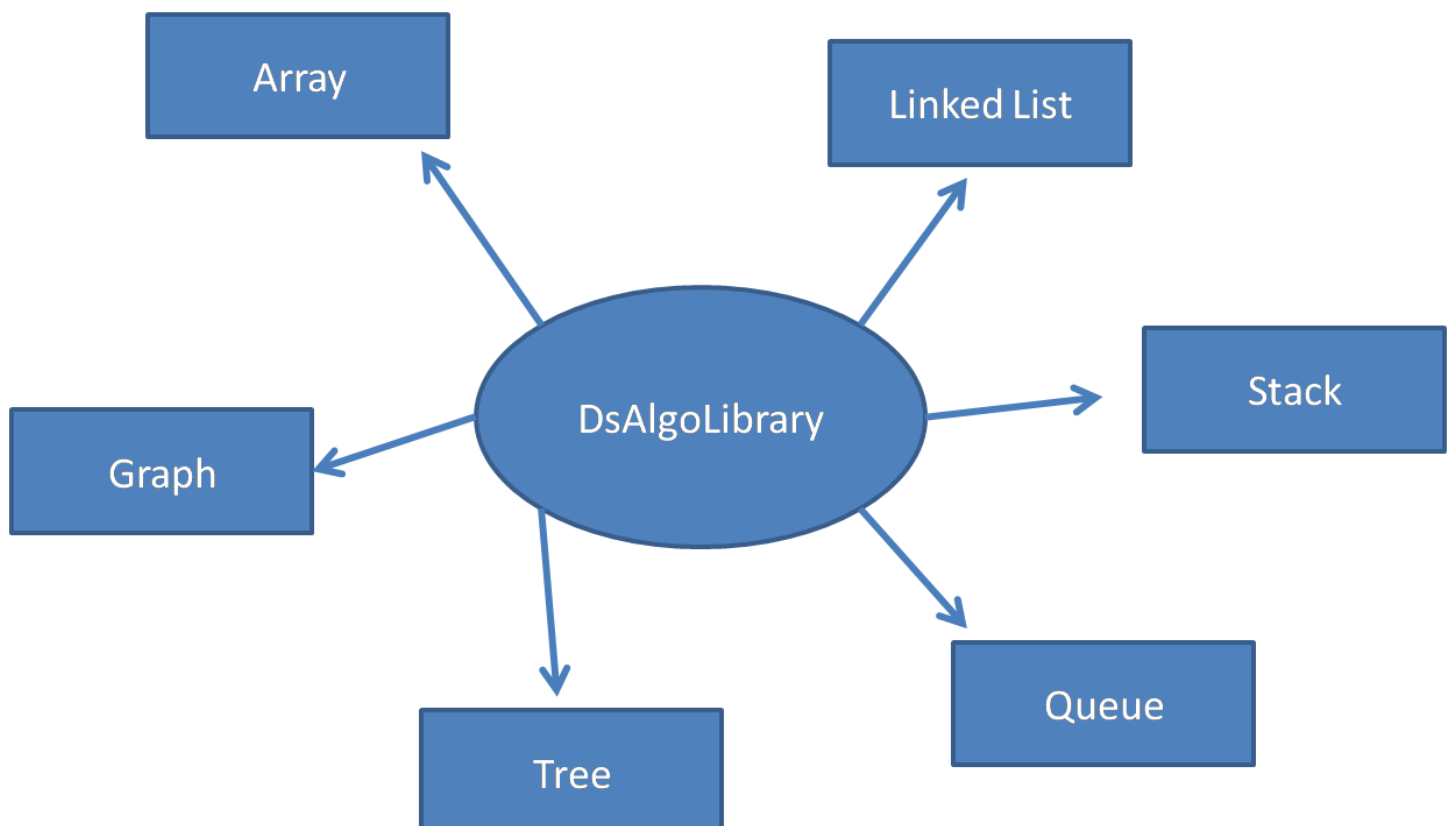
The design should have all the components like data structures, modules, and external libraries or namespace, etc.

5. **Maintainability:**

A good software design should be easily amenable to change whenever a change request is made from the user side.

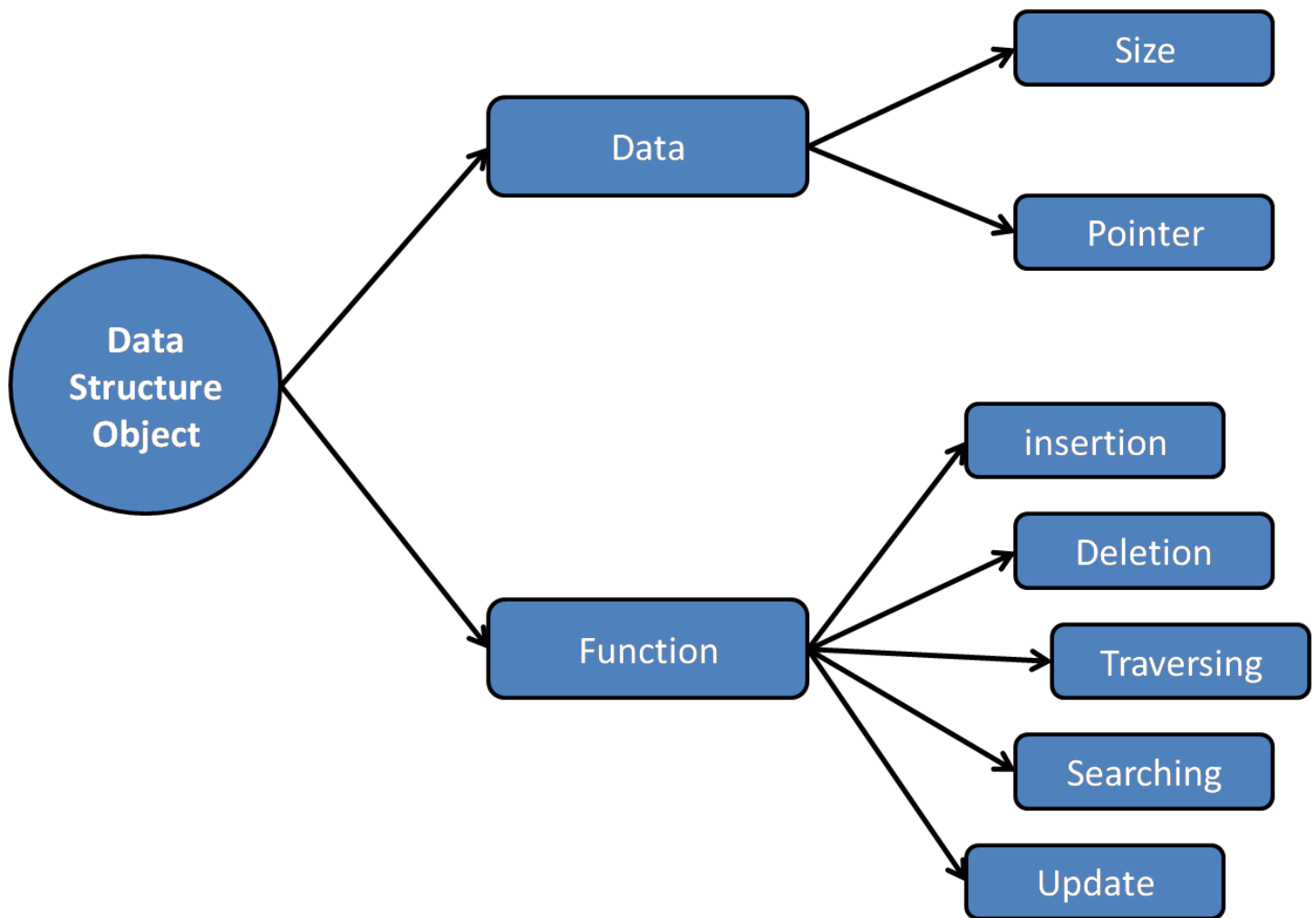
Library Diagram

Data Structure & algorithm Library should have 6 components that comprise all types of data structure and their functionality. Each component should have it's own functionality and algorithm implementation using C++.



Object Diagram of Data Structure

Object diagram of a data structure comprises the functionality and data of its object. That represents each entity that is being created for storing the data in the memory.



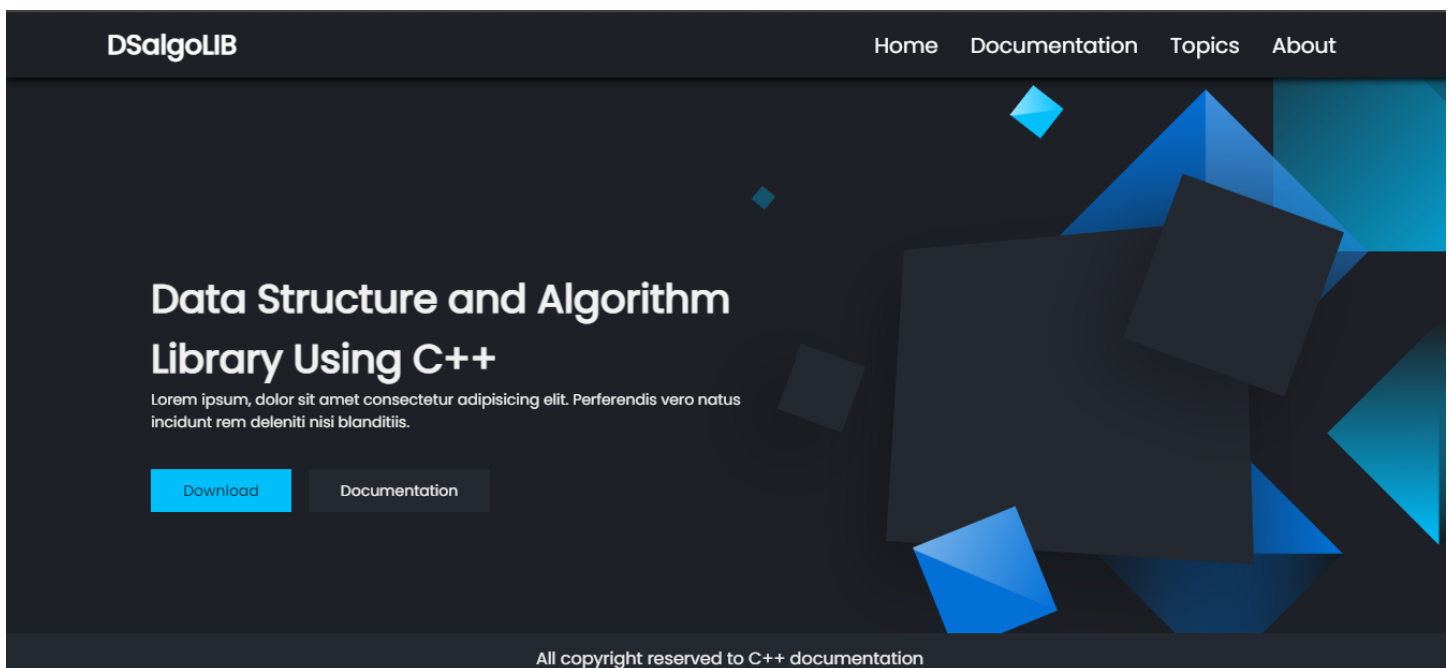
Library/Software Mapping

Following are the steps that we will take in consideration when mapping dsalgo library in any Client Driver Project using any software or code IDE.

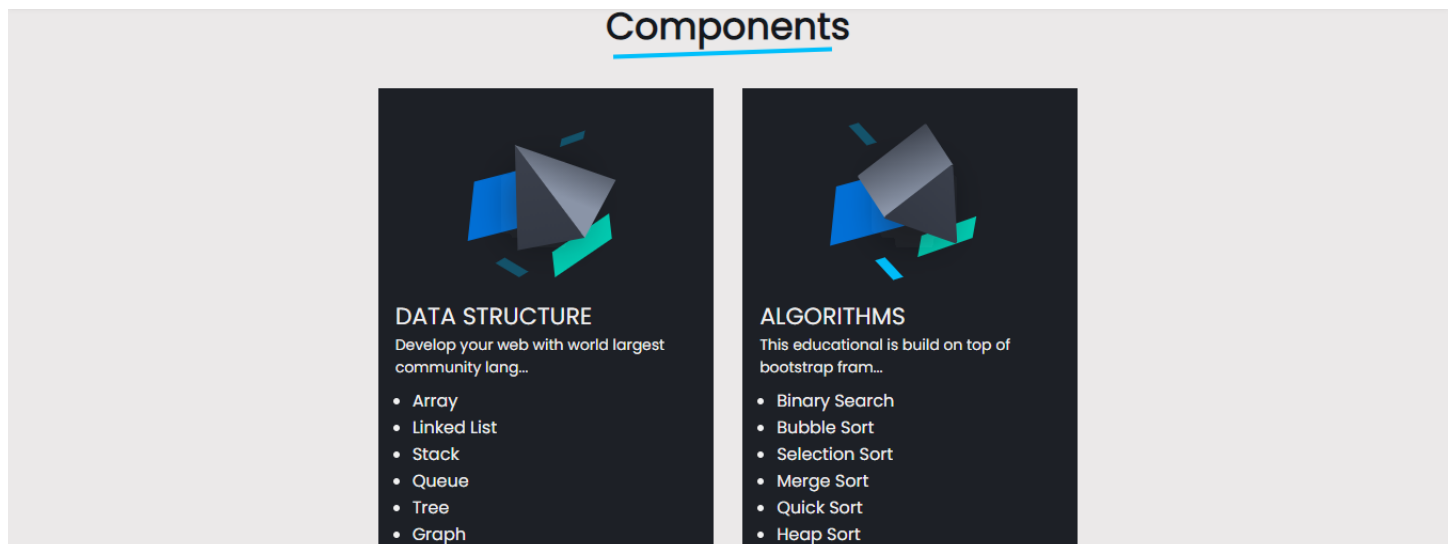
- Select the compiler on which you will have to work. You will have to take care of the version of any compiler as well as other dependencies mapping.
- Tell the compiler where to find the header and library files. Until you will not provide the specific path of the library and header file you will have issues to run the library function on the Driver Client application.
- Tell the linker used with compiler that which library is using your Driver Client Application
- Include header files on your Client application /project and make sure the project can find DLLs.

User Interface

The interface is user friendly and easy to use, you do not need to have any technical knowledge to download the library code in whatever system you are running your code. You simply hit the download button and your download will start automatically.



There are several components in data structure and algorithms, such as arrays, linked list, binary search, bubble sort etc. Well documentation for each component and it's parts are written on the website. User can easily navigate between components and its functionality.



In your Code Block IDE you can download and set up the library in a few simple steps. and Start integrating it with your Driver Application.

DS ALGO C++ LIBRARY

1. Installation

- Getting Started
 - Download
 - Setup

2. Data Structure

- Array
- Linked List
 - Introduction
 - Creating a Linked List
 - Insertion at Beginning
 - Insertion at End
 - Insertion after a Node
 - Deletion From Beginning
 - Deletion From End
 - Deleting a Specific Element

Download

- You can download dsalolib on various Windows versions.
- Dsalolib can be used on compiler have support for **g++ to follow C++11 ISO (-std=c++11)** or any higher version.

[Download dsalolib lib here](#)

Setup:-

Note:- The following setup is only given for CODE BLOCK C++ IDE.

Step:1-
First create a console based driver application in which you are going to use the data structure and algorithm.

Step:2-
Extract the download file of DSALGOLIB and configure the following setting for the

For creating any data structure and starting applying multiple functional algorithms on them, you can go through the documentation file.

Documentation file gives you the detail explanation of each function nomenclature like what type of parameter it takes and what type of value it return.

DSalgoLIB[Home](#)[Documentation](#)[Topics](#)[About](#)

DS ALGO C++ LIBRARY

1. Installation

- Getting Started
 - Download
 - Setup

2. Data Structure

- Array
- Linked List
 - Introduction
 - Creating a Linked List
 - Insertion at Beginning
 - Insertion at End
 - Insertion after a Node
 - Deletion From Beginning
 - Deletion From End
 - Deleting a Specific Element

Creating A Linked List

For creating a linked list using this library you just need to follow the given code:-

Step:1- Include Header files and Use name space as follow.

```
#include "linkedlistlib.h";  
using namespace std;  
using namespace LinkedList;
```

Step:2- Create object of the List class as follow

```
int main(void){  
    List <int> l1; // you can pass here the type of data on which you are working.  
}
```

All copyright reserved to C++ documentation

Section-4

Development

Core of DSALGO Library

Data Structure & Algorithm Library is developed using C++ Linked Library. In C++ we have two types of libraries.

1. Static Linked Library
2. Dynamic Linked Library

DSALGO library is developed using the concept of Dynamic Linked Library.

Dynamic Linked Library:-

Dynamic linking is a mechanism that links applications to libraries at run time. The libraries remain in their own files and are not copied into the executable files of the applications. DLLs link to an application when the application is run, rather than when it is created. DLLs may contain links to other DLLs.



























Advantages of DLL:-

1. Uses fewer resources
2. Promotes modular architecture
3. Aid easy deployment and installation

File Structure of Library

DSALGO library consist of multiple file with different extensions, each file having different data and different functionality to operate on the data inserted by the user.

File Structure also contains the Dev-c++ project itself and other dependent file with (.o,.dll,.a,.def,.layout,.h,.cpp etc) extension files.

	arraylib	03-Sep-21 12:43 PM	C++ Source File	8 KB
	arraylib	03-Sep-21 12:43 PM	C Header File	2 KB
	arraylib.o	03-Sep-21 12:43 PM	O File	3 KB
	DS_ALGO_TEMPLATE_LIBRARY	03-Sep-21 12:46 PM	Dev-C++ Project ...	3 KB
	DS_ALGO_TEMPLATE_LIBRARY.dll	03-Sep-21 12:43 PM	Application extens...	1,334 KB
	DS_ALGO_TEMPLATE_LIBRARY.layout	03-Sep-21 12:46 PM	LAYOUT File	1 KB
	dsalolib	03-Sep-21 12:46 PM	C Header File	1 KB
	graphlib	03-Sep-21 10:04 AM	C++ Source File	1 KB
	graphlib	03-Sep-21 10:05 AM	C Header File	1 KB
	graphlib.o	03-Sep-21 10:06 AM	O File	3 KB
	libDS_ALGO_TEMPLATE_LIBRARY.a	03-Sep-21 12:43 PM	A File	3 KB
	libDS_ALGO_TEMPLATE_LIBRARY.def	03-Sep-21 12:43 PM	DEF File	1 KB
	linkedlistlib	06-Aug-21 10:41 A...	C++ Source File	5 KB
	linkedlistlib	06-Aug-21 10:41 A...	C Header File	1 KB
	linkedlistlib.o	06-Aug-21 10:42 A...	O File	3 KB
	Makefile.win	03-Sep-21 12:43 PM	WIN File	2 KB
	queuelib	06-Aug-21 10:39 A...	C++ Source File	3 KB
	queuelib	06-Aug-21 10:36 A...	C Header File	1 KB
	queuelib.o	06-Aug-21 10:42 A...	O File	3 KB
	stacklib	06-Aug-21 10:38 A...	C++ Source File	2 KB
	stacklib.cpp	06-Aug-21 9:53 AM	C++ Source File	1 KB
	stacklib	06-Aug-21 10:39 A...	C Header File	1 KB
	stacklib.o	06-Aug-21 10:42 A...	O File	3 KB
	treelib	06-Aug-21 9:49 AM	C++ Source File	5 KB
	treelib	06-Aug-21 9:49 AM	C Header File	2 KB
	treelib.o	06-Aug-21 10:42 A...	O File	3 KB

In the dsalgo library there are 13 files which are developed explicitly apart from library files. These files contain the actual code that performs the operation of a data structure. Following are the files:-

- arraylib.h and arraylib.cpp
- stacklib.h and stacklib.cpp
- queuelib.h and queuelib.cpp
- linkedlistlib.h and linkedlistlib.cpp
- treelib.h and treelib.cpp
- graphlib.h and graphlib.cpp
- at the end dsalgotlib.h

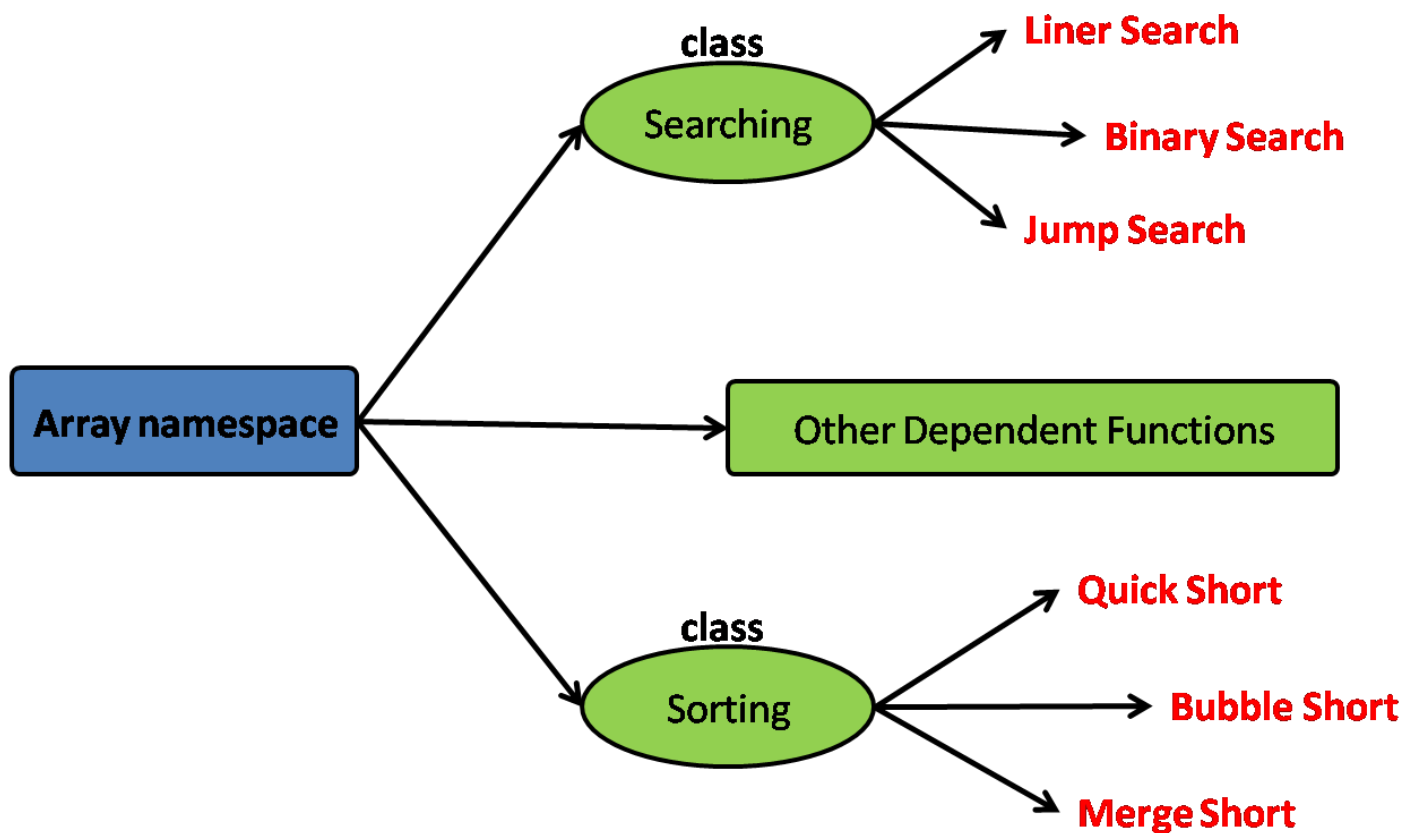
One by one we will see how these files work and how the code base is implemented.

Array's Files

Array's files implement all the operation of arrays such as insertion, deletion, traversing, searching, sorting etc.

There are two file of array data structure:-

1. `arraylib.h` :- `array.h` file is the header file for the array data structure. Including this file in your code will allow you to access all the operations of the array.
2. `arraylib.cpp` :- `array.cpp` file defines all the functionalities which are declared in the header files `array.h`.



Array.h file source code and It's explanation:-

```
#ifndef _arraylib_H_

#define _arraylib_H_


//predefined header files
#include<bits/stdc++.h>

using namespace std;


#if BUILDING_DLL

#define DLLIMPORT __declspec(dllexport)

#else

#define DLLIMPORT __declspec(dllimport)

#endif
```

Above code follows the basic structure of a DLL(Dynamic linked library). Which uses the functionality of a preprocessor to make decisions when which code should be run. It can be either import of DLL or export.

It also included all the necessary header files that are compulsory for a C PLUS PLUS code.

Searching class :-

Searching class of arrays namespace consists of multiple searching algorithms that can be used on the bases of type and size of data.

You can choose the best suited searching function for your application.

```
//Searching class
```

```
template<class A>
```

```
class Searching{
```

```
private:
```

```
    A *arrPtr; int size;
```

```
public:
```

```
    Searching(A arr[], int s);
```

```
    A linear_search(A item);
```

```
    int binary_search(int left, int right, A item);
```

```
    A jump_search(A item);
```

```
    A interpolation_search(int lower, int upper, A item);
```

```
};
```

Sorting Class :-

Sorting class of array namespace consist of multiple sorting algorithms that you can choose as per the pattern of data that is being used by your application.

```
//Sorting Class
```

```
    template <class A>
```

```
    class Sorting{
```

```
        private:
```

```
            A *arrPtr; int size;
```

```
        public:
```

```
            Sorting( A arr[], int s);
```

```
            A * bubble_sort();
```

```
            A * insertion_sort();
```

```
            A * selection_sort();
```

```
            A * merge_sort(int begin, int end);
```

```
            A * quick_sort(int low, int height);
```

```
            A * heap_sort();
```

```
            void display_array();
```

```
    };
```

Linked List's Files

Linked List's files implement all the operations of linked list such as insertion at beginning, insertion at end, deletion, traversing, searching, sorting etc.

There are two file of stack data structure:-

1. `linkedlistlib.h` :- `linkedlistlib.h` file is the header file for the linked list data structure. Including this file in your code will allow you to access all the operations of the linkedlist.
2. `linkedlistlib.cpp` :- `linkedlistlib.cpp` file defines all the functionalities which are declared in the header files `linkedlistlib.h`.

Linkedlistlib.h file source code and explanation:-

```
namespace LinkedList{  
    //defining class for performing all operation on singly linked list  
    template <class L>  
    class List{  
    public:  
        struct node{  
            L data; struct node *link;  
        };  
        List();  
        //get list function for returning node;  
        typename List<L>::node *
```

```
        get_list();  
  
        bool insertion_at_beginning(L value);  
  
        bool traversing();  
  
        bool insertion_at_end(L value);  
  
        bool insertion_after_node(L key_value, L value);  
  
        bool deletion_from_end();  
  
        bool deletion_from_beginning();  
  
        bool deletion_specific_element(L Key_value);  
  
        int linear_search(L value);  
  
        bool bubble_sort();  
  
    private:  
  
        node * header;  
  
};  
  
}
```

In the above code List is the class of LinkedList namespace. Each function of data manipulation in the List class has its own algorithm for performing the operation.

Stack's files

Stack's files implement all the operations of Stack operation such as push, pop, peak, traverse, searching, sorting etc.

There are two file of stack data structure:-

1. `stacklib.h` :- `stacklib.h` file is the header file for the stack data structure. Including this file in your code will allow you to access all the operations of the stack.
2. `stacklib.cpp` :- `stacklib.cpp` file defines all the functionalities which are declared in the header files `stacklib.h`.

`stacklib.h` file source code and explanation:-

```
namespace LinkedListStack{  
  
    template<class S>  
  
        class Stack{  
  
            public:  
  
                struct Node{  
  
                    S data; struct Node *link;  
  
                };  
  
                Stack();  
  
                typename Stack<S>::Node *  
  
                get_stack();  
}
```

```
bool push(S value);  
void traversing();  
S pop();  
S peek();  
int linear_search(S item);  
void bubble_sort();  
private:  
Node * top;  
};  
}
```

Above written code shows what will be the structure of the stack object and how many operations you can perform on stack. It also shows the syntax of functions that are being used in the class.

Stack is implemented using the concept of linked list. Top is the pointer that decided weather there is space or any element is for push and pop operation in the stack

Queue's files

queue's files implement all the operations of Queue operation such as queue, dequeue, traverse, searching, sorting etc.

There are two file of queue data structure:-

1. `queuelib.h` :- `queuelib.h` file is the header file for the queue data structure. Including this file in your code will allow you to access all the operations of the queue.
2. `queuelib.cpp` :- `queuelib.cpp` file defines all the functionalities which are declared in the header files `queuelib.h`.

`queuelib.h` file source code and explanation:-

```
namespace LinkedListQueue{  
  
    template <class Q>  
  
    class Queue{  
  
    public:  
  
        struct node{  
  
            Q data; struct node *link;  
  
        };  
  
        Queue();  
  
        //get list function for returning node;  
  
        typename Queue<Q>::node *  
  
        get_queue();  
}
```

```
void insertion(Q value);  
  
void traversing();  
  
Q deletion();  
  
int linear_search(Q item);  
  
void bubble_sort();  
  
private:  
  
node * front, *rear;  
  
};  
  
}
```

Queue class is defined inside the LinkedListQueue namespace. The Queue class uses two main pointers (front and rear) for performing operations on Queue data structures. Queue is a structure node which has two sections (1) Data and (2) Address of the next node. Queue class defines all the operations that can be performed on queue data structure.

Tree's Files

The tree's files implement all the operations of BinaryTree such as insertion, deletion, traverse(preorder, postorder, inorder), searching, sorting etc.

There are two file of tree data structure:-

3. `treelib.h` :- `queuelib.h` file is the header file for the Tree data structure. Including this file in your code will allow you to access all the operations of the Tree.
4. `teelib.cpp` :- `queuelib.cpp` file defines all the functionalities which are declared in the header files `treelib.h`.

treelib.h file source code and explanation:-

```
namespace BinaryTree{  
    template<class T>  
    class Tree{  
    private:  
        struct TreeNode{  
            T value; TreeNode *left, *right;  
        };  
        TreeNode *root;  
        void insert(TreeNode *&, TreeNode *&);  
        void destroy_sub_tree(TreeNode *);  
        void delete_node(T, TreeNode *&);  
    }  
};
```

```
void make_deletion(TreeNode *&);

void display_inorder(TreeNode *);

void display_preorder(TreeNode *);

void display_postorder(TreeNode *);

int count_nodes(TreeNode *&);

public:

    Tree(){ root = NULL; }

    ~Tree(){ destroy_sub_tree(root); }

    void insert_node(T);

    bool search_node(T);

    void remove(T);

    void display_inorder(){ display_inorder(root); }

    void display_preorder(){ display_preorder(root); }

    void display_postorder(){ display_postorder(root); }

    int num_nodes();

};

}
```

Above written code shows what will be the structure of the BinaryTree object and how many operations you can perform on BinaryTree. It also shows the syntax of functions that are being used in the class.

BinaryTree is implemented using the concept of Double linked list. Left and right are the two pointers which manage the functionality of a BinaryTree.

Graph's Files

The Graph's files implement all the operations of Weighted Graph such as adding an edge, traversing graph etc.

There are two file of tree data structure:-

1. graphlib.h :- queue.h file is the header file for the Graph data structure. Including this file in your code will allow you to access all the operations of the Graph.
2. graphlib.cpp :- graphlib.cpp file defines all the functionalities which are declared in the header files graphlib.h.

treelib.h file source code and explanation:-

```
namespace WeightedGraph{  
  
    template<class G>  
  
    class Graph{  
  
    private:  
  
        vector<pair<int,int> > * adjancy_list;  
  
        int vertices;  
  
    public:  
  
        Graph(int V){  
  
            vertices = V;  
  
            adjancy_list = new vector< pair<int,int> >[vertices];  
  
        }  
  
        void add__edge(int v1,int v2, int wt);  
  
        void display_graph(); }; }
```

DSALGOLIB.h File

The DSALGOLIB.h file consists of all the header files which we discussed just above.

dsalolib.h file

```
#include "arraylib.h"
```

```
#include "linkedlistlib.h"
```

```
#include "stacklib.h"
```

```
#include "queuelib.h"
```

```
#include "treelib.h"
```

```
#include "graphlib.h"
```

If you don't want to take the headache of adding those multiple files multiple times in your project or application then you can just include the DSALGOLIB.h file that includes all the required .h file for implementation of data structure and algorithms.

Section-5

Testing

Unit Testing

Unit testing is undertaken when a function has been created and successfully reviewed.

In order to test a single function, we need to provide a complete environment i.e. besides the function we would require.

- The procedures belong to other functions that the function under test calls.
- Non local data structures that module accesses.
- A procedure to call the functions of the module under test with appropriate parameters

Unit testing was done on each and every function that is described under function description.

Integration Testing

In this type of testing, we test multiple classes inside a file and we test various integration of the project module by providing the input.

The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module. It is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers.

System Testing

The aim of the system testing process was to determine all defects in our project.

The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not.

In this the complete library is checked, it is defined as **testing of a complete and fully integrated software product**.

This testing falls in black-box testing wherein knowledge of the inner design of the code is not a prerequisite and is done by the testing team.

Section-6

Implementation

1. **Library Setup :-** You can set up the dsalgo library on your Driver Client Project easily using Code Block IDE within a few of the steps.

Step:1-

First create a console based driver application in which you are going to use the data structure and algorithm.

Step:2-

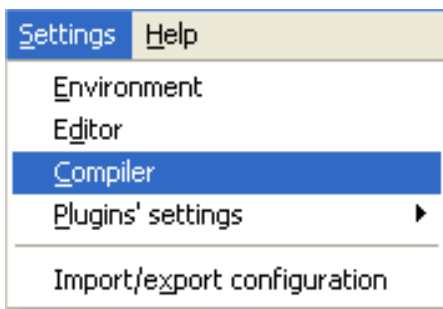
Extract the download file of DSALGOLIB and configure the following setting for the Driver Project.

Step:3 -

Go to **setting -> compiler -> compiler Setting**. Select the version of your g++ compiler . This library is developed in **g++ to follow C++11 ISO (-std=c++11)** so for best performance of the library functionality you can select the above compiler settings.

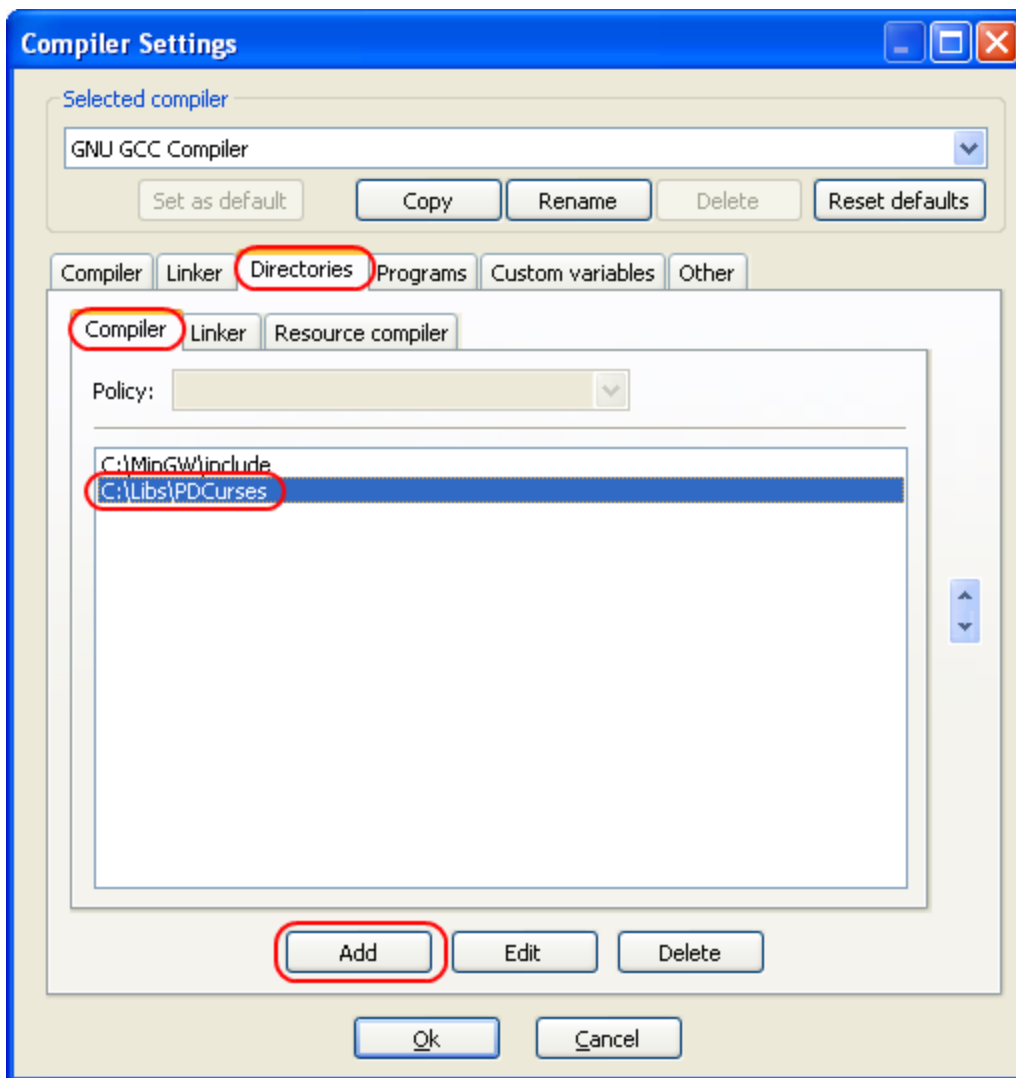
Step:4 -Tell the compiler where to find headers and library files.

We are going to do this on a global basis so the library will be available to all of our projects. Consequently, the following steps only need to be done once per library. A) Go to the “Settings menu” and pick “Compiler”.

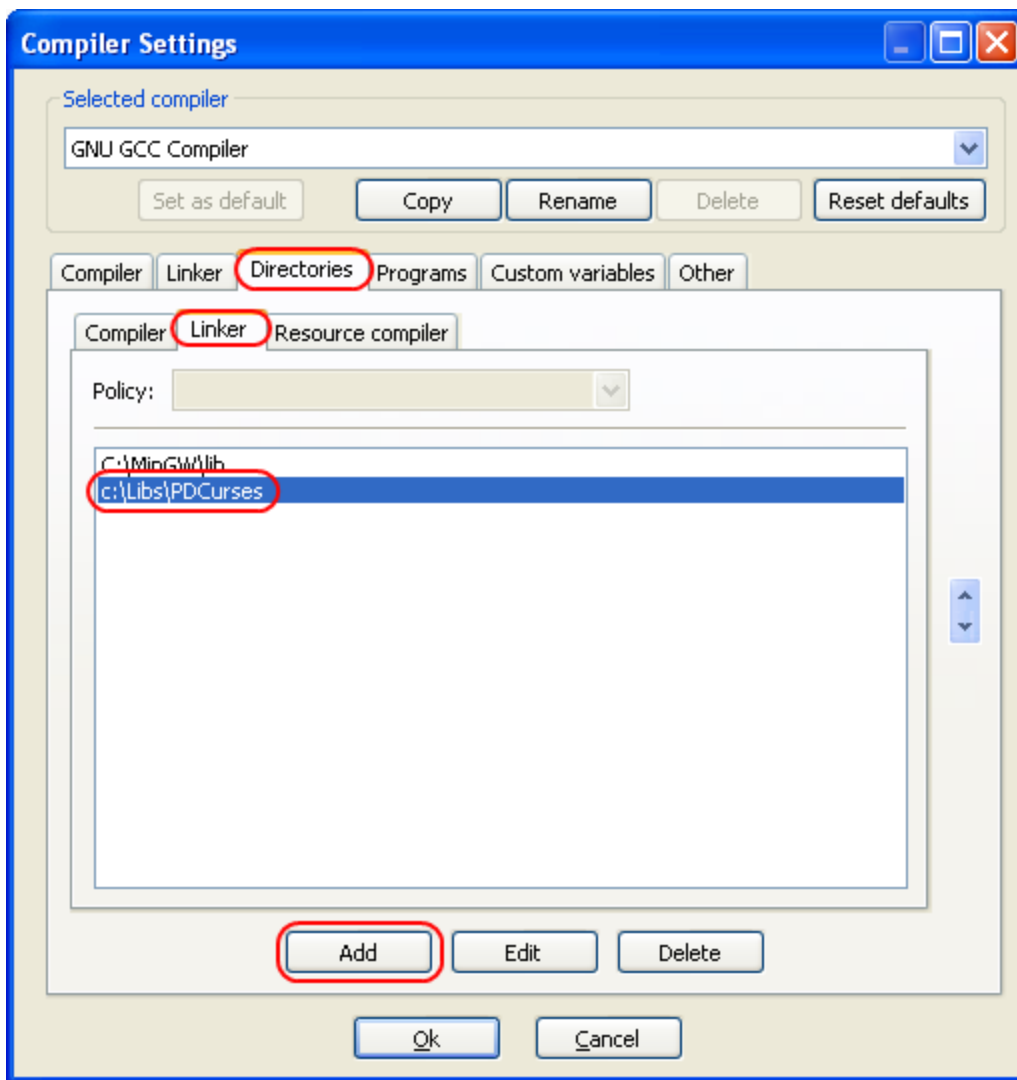


B) Click the “Directories” tab. The compiler tab will already be selected for you.

C) Press the “Add” button, and add the path to the .h files for the library.



D) Click the “Linker” tab. Press the “Add” button, and add the path to the .lib files for the library. If you are running Linux and installed the library via a package manager, make sure /usr/lib is listed here.

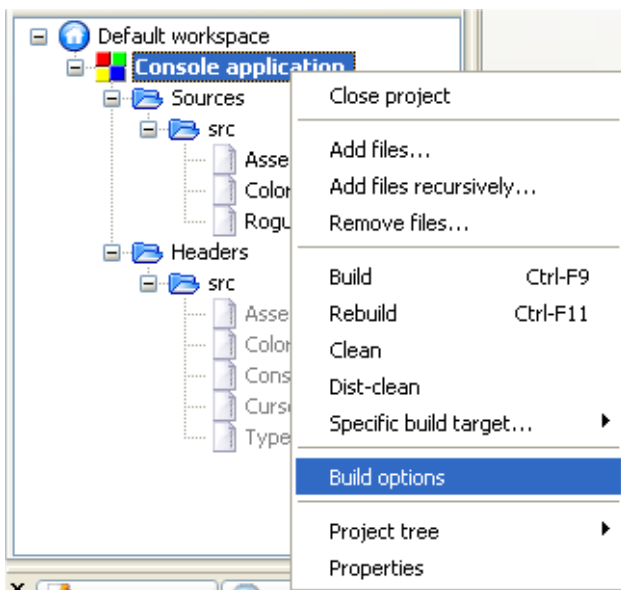


E) Press the “OK” button.

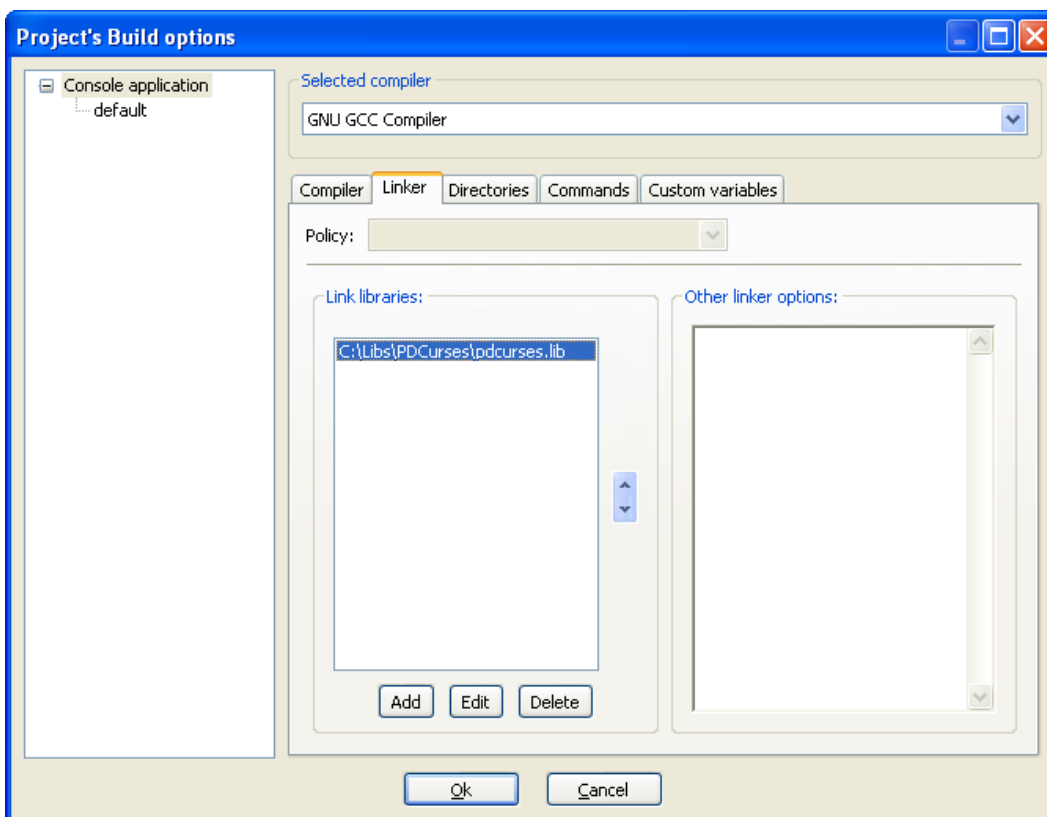
Step 5 --Tell the linker which libraries your program is using

For step 5, we need to add the library files from the library to our project. We do this on an individual project basis.

A) Right click on the bolded project name under the default workspace (probably “Console application”, unless you changed it). Choose “Build options” from the menu.



B) Click the linker tab. Under the “Link libraries” window, press the “Add” button and add the library you wish your project to use.



C) Press the “OK” button

Steps 6 and 7 -- #include header files and make sure project can find DLLs Simply #include the header file(s) from the library in your project.

2. Documentation of Library:-

- A website is developed using the most trending framework Django(Python) and sqlite database for Documentation of libraries.
- Documentation is organized in accordance with target audience needs.
- Documentation of DSALGO Library is clean and understandable it's not unambiguous
- There is no repetition in the documentation. That gives clarity of the concepts. Industry standards should be used
- The Developer Team will try to keep updating the documentation of the library for ease of user.
- Access to the website can be done using the below url.

dsalgotlib.pythonanywhere.com

References

- “Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles” by Narasimha Karumanchi.
- Fourth Edition Data Structures and Algorithm Analysis in C++ Mark Allen Weiss Florida International University
- “GeeksForGeeks” computer science learning portal.
- “Tutorialspoint” Computer Science Tutorial Learning Portal.