Data Structure and Algorithm Analysis

Course Description:

Solving computational problems that involve manipulating collections of data, study a core set of data abstractions, data structures and algorithms that provide a foundation for writing efficient programs.

Number of units laboratory: 2 units

Number of contact hours per week: 6 hours per week

Prerequisite: Object Oriented Programming

Course Outline

- Stack Abstract Data Type
- Queue Abstract Data Type
- Link List: Singly Link List, Doubly Link List
- Tree ADT
- Binary Search Tree
- AVL Tree
- Heaps
- Basic Algorithm Analysis
- Algorithm Strategies
- Classic Algorithms for Common Tasks
- Parallel Algorithms and Multithreading
- Algorithmic Complexity
- Scheduling Algorithms
- Basic Computability Theory

Application Programming Language: C++

Chapter 1

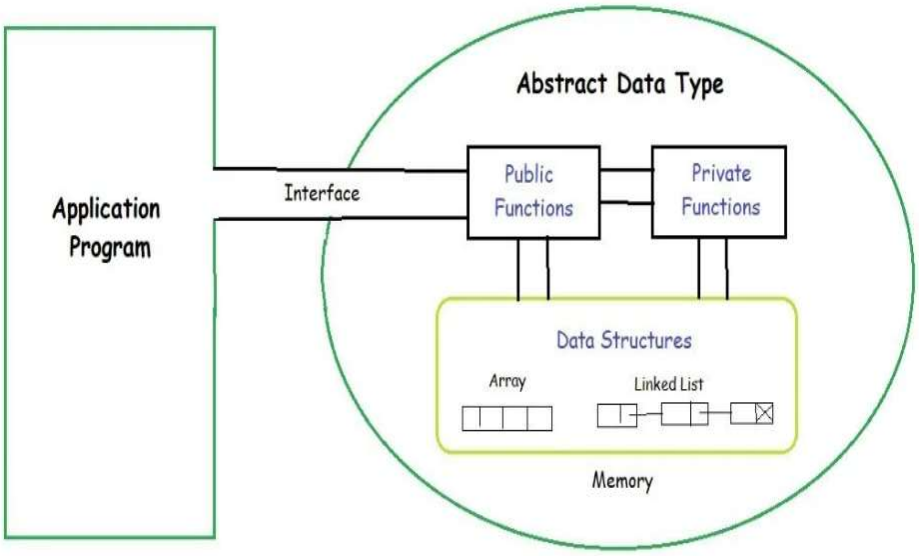
Stack Abstract Data Type

Learning Objectives:

1. To learn about the stack ADT and its different operations
2. To be able to implement stack ADT using CPP.
3. To solve problems that requires stack AD

Abstract Data Type (ADT)

An ADT is a type or class for objects whose actions is defined by a set of value and a set of operations. ADTs are only focused on what operations are to be performed and does not focus on how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called abstract because it gives an implementation-independent view. The process of providing only the essentials and hiding the details is known as abstraction.

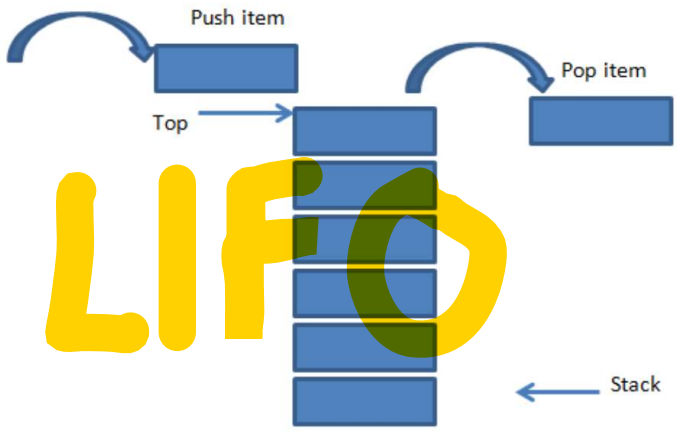


The one who uses the data type does not need to know how that data type is being implemented. ADTs are like the black box in a machine. These black box hides the inner structure and design of the machine. ADT is the same in concept with these black box.

STACK ABSTRACT DATA TYPE

Stack is a fundamental data structure which is used to store elements in a linear way. Stack can be found in our daily lives. One good analogy of a stack, is the stacking of plates or books. You stack them starting from the bottom to top.

Illustration of a stack.



As shown on the figure, there is a pile of information stacked on top of each other. If we want to add another information, then we add it on top of the stack. To remove information from a stack, this is also done at the same end of the stack.

Applications of a Stack:

1. Page-visited history in a web browser.
2. Undo sequence in a text editor.
3. Storing and retrieving jobs in your operating system.

Stack Declaration

The instruction blow shows a how stack is created or declared.

```
class Stack {
        private:
                int top;
                int ele[MAX];
        public:
                Stack();
                int isFull();
                int isEmpty();
```

```
        void push(int item);
        int pop(int *item);
};
```

Basic Operations of a Stack

A stack follows the Last In First Out (LIFO) property. This means that to remove an element of a stack, the element that is to be removed was the who one came in last. To add an element on the stack, the new element shall be placed on top of the last information who came in last.

1. **push()** – stores a new element in your stack. Exception: you cannot push an element if your stack is full.

   push() implementation:

```
void stack::push(int item) {
     if(isFull)) {
          cout<<"Stack is Full";
          return;
     }
     top++;
     else[top] = item;
     cout<<"Inserted data"<<item;
}
```

   Exception: isFull() - checks if the stack is full.

   isFull() implementation:

```
int Stack:: isFull() {
   int full = 0;
   if(top == MAX-1)
      full = -1;
   return full;
}
```

2. pop() –removes an element from your stack. Exception: you cannot pop an element if the stack is empty.

   pop() implementation:

```
int Stack:: pop (int *item)
{
    if(isEmpty( )) {
       cout<<"\nStack Underflow";
       return -1;
    }
*item = ele[top--]
 return 0;
}
```

3. isEmpty() – check if the stack is empty.
4. isFull() – checks if the stack is full
5. printStack() – prints the elements of your stack.
6. top() – checks and displays the element found on the top of the stack
7. bottom()- checks and displays the element found on the bottom of the stack.

Assignment

Write a program that will n numbers of integer. Your program will store, remove, display the top, bottom, the stack and reverses the order of the elements of your stack

Illustration:

```
Enter n number of inputs; 5
Stack options:
      [A] push()
      [B] push()
      [C] top()
      [D] bottom()
      [E} reverse()
```

Note: pm me for additional information regarding this assignment.