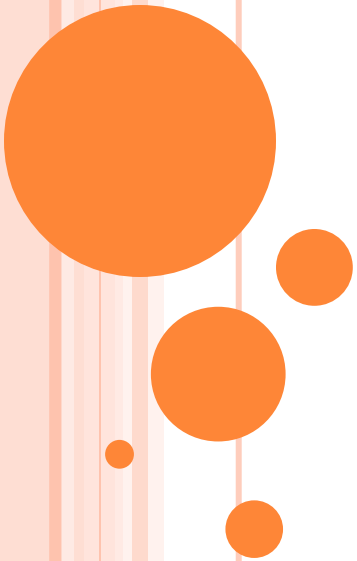


DATA STRUCTURES LAB 2

Stacks

Outline

- Data Structures Introduction
- Stack ADT
- Applications of stack
- Implementation of stack
- Prelab questions
- Exercise

Data Structures Introduction

- Date: are simply a value are set of values of different type which is called data types like string, integer, char etc.
- Structure: way of organizing information, so that it is easier to use.
- In simple words we can define data structures:

  It's a way of organizing data in such a way so that data can be easier to use.
- A data structure is a particular way of organizing data in a computer so that it can be used efficiently.
- A scheme for organizing related pieces of information.

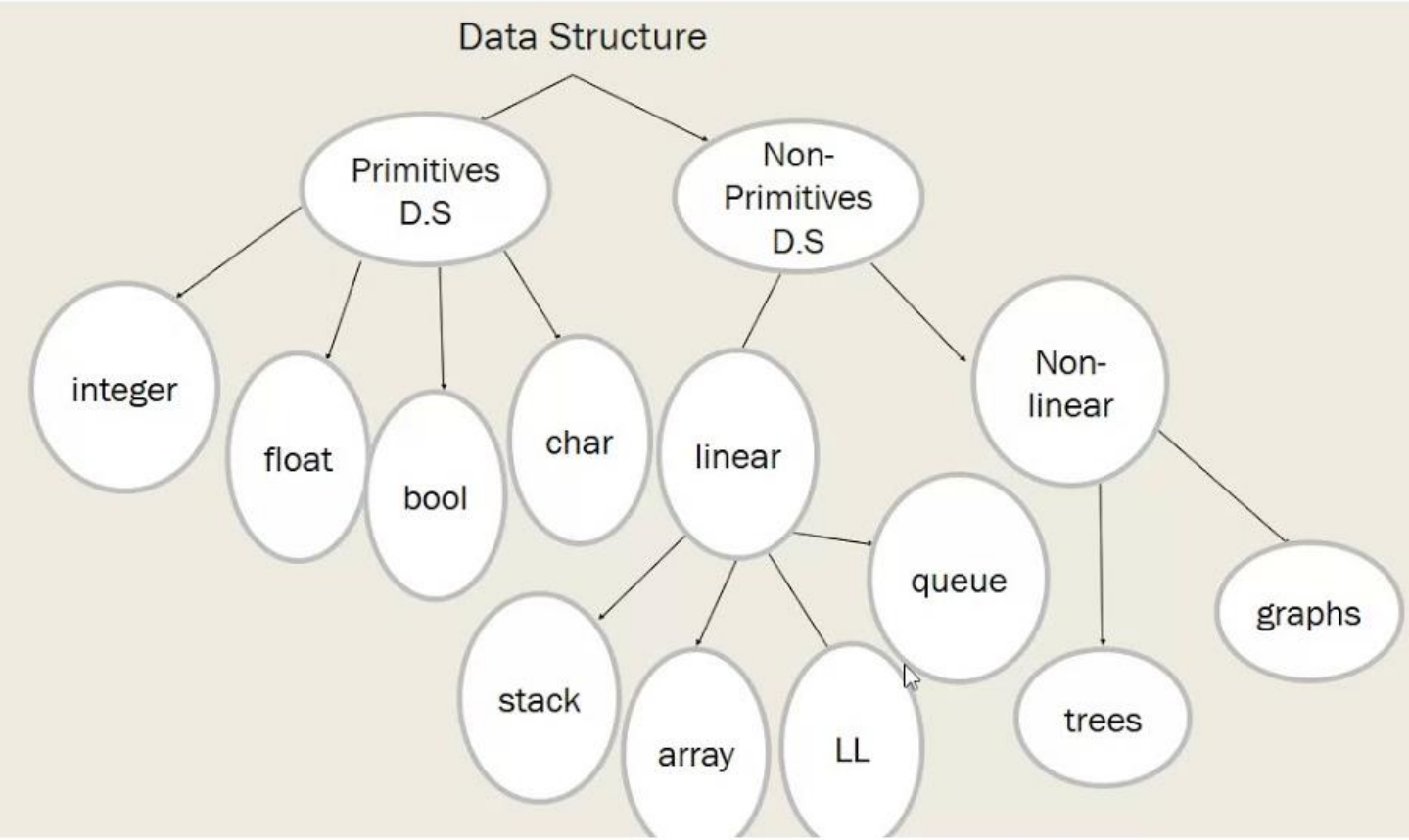
Why Data Structures?

- Human requirement with computer are going to complex day by day. To solve the complex requirements in efficient way we need this study.
- Provide fastest solution of human requirements.
- Provide efficient solution of complex problem.

  --- Space

  --- Time

Classification of Data Structure



Stack

- Stack is formally defined as an ordered list, in which insertion and deletion are done at one end, called top.
- One more name used for stacks is LIFO (last in first out) or FILO (first in last out) lists because the last element inserted is the first one to be deleted.
- When we add an item to a stack, we say that we push it onto the stack and when we delete an item we say that we pop it from the stack.
- We might attempt to pop an item from an empty stack (underflow) or push an item onto a full stack (overflow). We shall regard such attempts as errors.

How stacks are used

Consider a working day in the office. Let us assume a developer is working on a long-term project. The manager then gives the developer a new task which is more important. The developer puts the long-term project aside and begins work on the new task. The phone rings, and this is the highest priority as it must be answered immediately. The developer pushes the present task into the pending tray and answers the phone. When the call is complete the task that was abandoned to answer the phone is retrieved from the pending tray and work progresses.

Applications:

- Some of the applications in which stacks play an important role:
  - Direct Applications:
    - Balancing of symbols
    - Infix-to-postfix conversion
    - Evaluation of postfix expression
    - Implementing function calls (including recursion)
    - Page-visited history in a Web browser [Back Buttons]
    - Undo sequence in a text editor.
    - Matching Tags in HTML and XML.
  - Indirect Applications:
    - Auxiliary data structure for other algorithms (e.g. tree traversal algorithms)
    - Component of other data structure (e.g. simulating queues)

Stack ADT

- Data
  - Space for storing elements
  - Top pointer
  - Size
- Operations
  - Push(element)
  - Pop()
  - Peek(index)
  - isEmpty()
  - isFull()
  - Display()

ALGORITHM TO INITIALIZE A STACK:

- Step1: Declare a structure with tag name stack and 3 fields
  1) Integer array pointer.
  2) Integer variable top.
  3) Integer variable size.
- Step 2: Define a pointer SP which points to stack make top=-1
- Step 3: End.

ALGORITHM FOR PUSH

- Step1: Check whether top equals maximum size of stack.

  If yes go to step3 else go to step2
- Step2: Increment top and assign S [top]=item. Go to step 4.
- Step3: Print “stack overflow”.
- Setp4: End

Algorithm for POP

- Step1: Check whether stack top is equal to -1. If yes go to step3 else go to step2.
- Step2: Assign S [top] to a variable X decrement top and return X. Go to step 4.
- Step3: Print “stack under flow”.
- Step4: End.

ALGORITHM FOR PEEK:

- Step1: Check whether stack top is equal to -1. If yes go to step3 else go to step2.
- Step2: Assign S [top] to a variable and return X. Go to step 4.
- Step3: Print “stack under flow”.
- Step4: End.

ALGORITHM FOR DISPLAY:

- Step1: Check whether stack top is equal to -1. If yes go to step3 else go to step2.
- Step2: Assign S [top] to a variable X decrement top and return X until stack top reaches to bottom of the stack
- Step3: Print “stack under flow”.
- Step4: End.

Exercise

- Write a menu driven C program that implements Stacks using arrays(implement the algorithms)

  a)create b)push c)pop d)peek e)display
- Given an expression string exp , write a program to examine whether the pairs and the orders of “{“,”}”,”(“,”)”,”[“,”]” are correct in exp.

  For example, the program should print true for exp = “[()]{}{[()()]()}” and false for exp = “[(])”
- Write a program to add individual digits of a given number using stack.
- Given a string str, write a program to print reverse of individual words using stack.

Prelab Questions

1. Define stack.
2. List out the applications of stack.
3. Can we delete any element from stack. Justify your answer.
4. How do you check empty condition and full condition of a stack.
5. Is it possible to implement 3 stacks in one array? If yes, how, if no , why.
6. What are the complexities for stack operations.