P1 Create a design specification for data structures explaining the valid operations that can be carried out on the structures.

1.1 definition

In computer science, an abstract data type is a theoretical data type that is largely defined by the operations and work on it and the limitations that apply. Professionals describe an abstract data type as a “mathematical model” for groups of data types, or as a “value with associated operations” that is independent of a particular implementation. In other words, we can say that abstract data types are the entities that are definitions of data and operations but do not have implementation details. In this case, we know the data that we are storing and the operations that can be performed on the data, but we don't know about the implementation details. The reason for not having implementation details is that every programming language has a different implementation strategy for example; a C data structure is implemented using structures while a C++ data structure is implemented using objects and classes.

1.2 ways to represent ADT

1.2.1 Stack

The stack abstract data type is constructed as an ordered collection of things, with items being added to and deleted from the “top” end. Stacks are arranged in a LIFO fashion. The operations on the stack are listed below:

isFull(): This is used to determine whether or not the stack is full.

isEmpty(): This is used to determine whether or not the stack is empty.

push(x): This is used to place x at the top of the stack.

pop(): This is used to remove one element from the stack’s top.

peek(): This is used to obtain the stack’s topmost member.

size(): This function is used to determine the number of elements in the stack.

1.2.2 LIst

The idea of position is the most significant in the context of lists. To put it another way, we consider the list to have a first element, a second element, and so on. As a result, define a list as a finite, ordered series of data elements. This is similar to the idea of a series in mathematics. In this definition, “ordered” indicates that each element has a place in the list. In this case, the term “ordered” does not imply that the list members are sorted by value. The data is usually kept in a list with a head structure that includes a count, pointers, and the location of the comparison function that is used to compare the data in the list. The data node carries a data structure pointer as well as a self-referential pointer to the next node in the list. On the list, you may do the following operations:

get(): Return an element from the list at a certain point in the list.

insert(): Add a new element to the list at any point.

remove(): In a non-empty list, remove the first occurrence of any element.

removeAt(): Remove the entry from a non-empty list at a given position.

replace(): A different element can be used to replace an element in any location.

size(): The number of elements in the list is returned.

isEmpty(): If the list is empty, return true; otherwise, return false.

isFull(): If the list is empty, return true; otherwise, return false.

1.2.3 queue

Queue is an abstract data structure, something similar to a queue in everyday life (queuing). Unlike the stack, the queue is open at both ends. One end is always used to insert data (also called rowing) and the other end is used to delete data (leave rows). The queue data structure follows the First-In-First-Out method, i.e. the data entered first will be accessed first.

enQueue(value): To add a new element to the queue.

deQueue(): To remove an item from a queue.

display(): To view the queue’s components

1.3 type of data structures

Data structure is a way to store, organize data, orderly, systematic, so that the data can be used efficiently.

Below are two concepts forming the foundation should a data structure:

Interface: Each data structure has an Interface. Interface performing a set of calculations that a data structure supports. An Interface that provides the list of arithmetic is supported, the type of parameters that they can accept and return type of this calculation.

Implementation (can be understood as the deployment): Provide the performing internal of a data structure. Implementation also provides the definition of the algorithm used in the calculation of the data structure.

Characteristics of a data structure

Correct: The implementation of structured data should implement the Interface of it correctly.

Complexity of time (Time Complexity): run-Time or execution time of the calculation of the data structure must be as small as possible.

The complexity of memory (Space Complexity): The memory usage of each calculation of the data structure should be as small as possible.

Why a data structure is needed ?

Today, the application of increasing complexity and amount of data is growing with more diverse. This appearance 3 big problems that every programmer face:

Search data: suppose there are 1 million of goods to be kept on in stock goods. And suppose there is an app need to search for a commodity. Then each when performing a search, this app will be looking for 1 goods in-1 million of the goods. As data grows, search will become increasingly slower and more expensive.

Speed processor: although the processor has very high speed, but it also has limits, and when your data at up to billions of records, then the processing speed will also no longer be fast again.

Multi requirements: When thousands of users and carry out a search on a Web Server, then even though the Web Server that fast to you then have to handle thousands of calculations at the same time is really very difficult.

To handle the above problem, the data structure is a great solution. Data can be organized in a data structure in a way to when performing searches for an element, then the requested data will be found immediately.

1.3.1 linear da Advantages of data structures

• Data structure allows to store information on the hard drive.

• provides a way to manage large sets of data such as databases or services indexing the internet.

• Have the need to design efficient algorithms not?

• allows secure storage of information on your computer. This information can then be used

• for later use and can be used by different programs. In addition, information is protected

• and should not be lost (especially if it was stored on magnetic tape).

• allows data to be used and disposed on system software.

• Allows data processing easier.

• Use the internet, we can access the data anytime from any machines that are connected (computer,

• computer, laptop, tablet, phone, v.v.)

Disadvantages of data structure

• Only the sophisticated user can make changes to data structures

• Any problems related to data structures all need to come to the aid of experts, which is the basic

• users can't help a structure

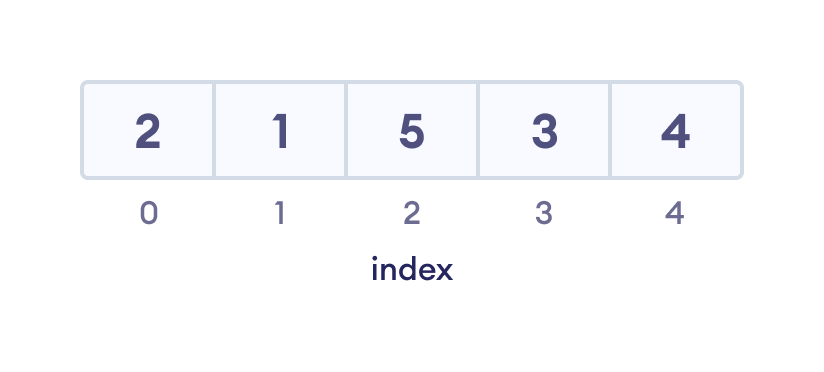
A data structure is considered linear if the data elements construct a linear list sequence. Elements are tied together and in a defined order. It consumes memory space linearly, data elements are required to store sequentially in memory. While implementing a linear data structure, the amount of memory required is previously declared. It doesn't make good use of memory and leads to wasted memory. The data element is accessed sequentially where only one can be accessed directly.

Examples included in linear data structures are arrays, stacks, queues, linked lists, etc. An array is a group of a certain number of identical elements or items. data. Stacks and queues are also an ordered set of elements like an array but there is a special condition where the stack is LIFO (Last First Out) order and the queue uses FIFO ( First in first) to insert and delete elements. A list can be defined as a collection of variable numeric data items.

**Popular linear data structures are:**

**Array Data Structure**

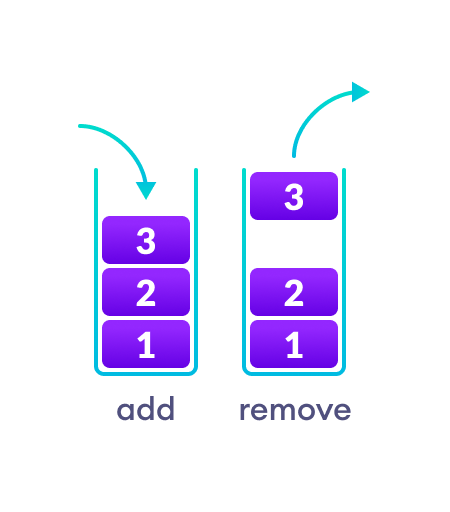
In an array, elements in memory are arranged in continuous memory. All the elements of an array are of the same type. And, the type of elements that can be stored in the form of arrays is determined by the programming language.



**Stack Data Structure**

In stack data structure, elements are stored in the LIFO principle. That is, the last element stored in a stack will be removed first.

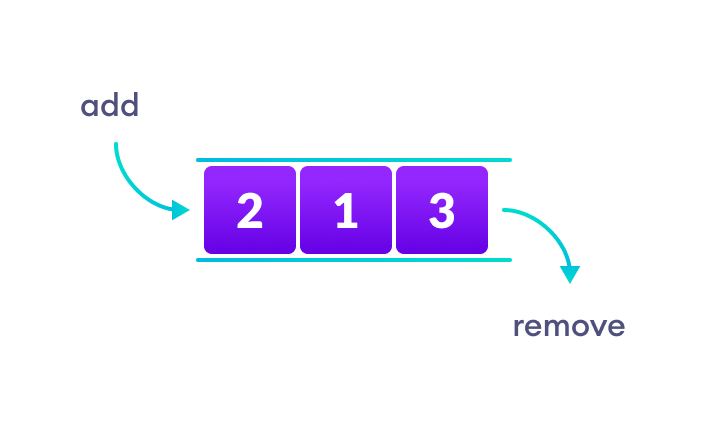
It works just like a pile of plates where the last plate kept on the pile will be removed first.



**Queue Data Structure**

Unlike stack, the queue data structure works in the FIFO principle where first element stored in the queue will be removed first.

It works just like a queue of people in the ticket counter where first person on the queue will get the ticket first.



**Linked List Data Structure**

In linked list data structure, data elements are connected through a series of nodes. And, each node contains the data items and address to the next node.



**Non linear data structures**

Non-linear data structures do not arrange consecutive data instead it is arranged in sort order. In it, data elements can be attached to more than one element representing a hierarchical relationship regarding the relationship between child, parent and grandparent. In a non-linear data structure, the transmission of data elements and insertion or deletion are not performed sequentially.

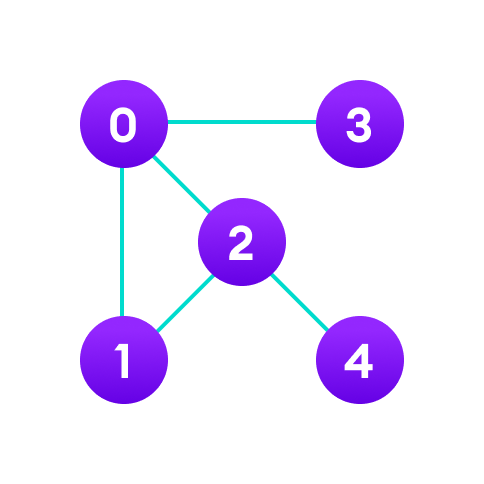
Non-linear data structures use memory efficiently and do not require prior memory declaration. There are two common examples of non-linear data structures - trees and graphs. A tree data structure organizes and stores data elements in a hierarchical relationship.

Unlike linear data structures, elements in non-linear data structures are not in any sequence. Instead they are arranged in a hierarchical manner where one element will be connected to one or more elements.

Non-linear data structures are further divided into graph and tree based data structures.

**1. Graph Data Structure**

In graph data structure, each node is called vertex and each vertex is connected to other vertices through edges.



**Popular Graph Based Data Structures:**

• Spanning Tree and Minimum Spanning Tree

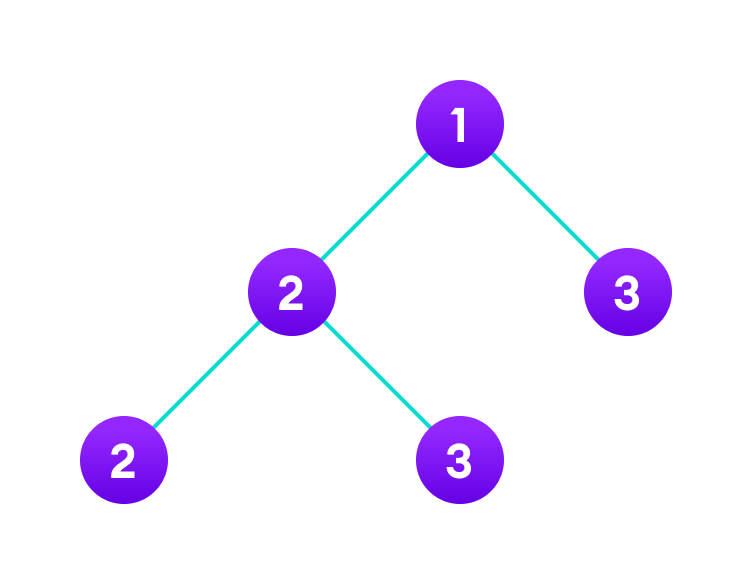
• Strongly Connected Components

• Adjacency Matrix

• Adjacency List

**Trees Data Structure**

Similar to a graph, a tree is also a collection of vertices and edges. However, in tree data structure, there can only be one edge between two vertices.



**Popular Tree based Data Structure**

• Binary Tree

• Binary Search Tree

• AVL Tree

• B-Tree

• B+ Tree

• Red-Black Tree

1.4 example

Data structure stack:

Stack is a data structure linear comply with specific order in which the activities

've made. Order may be LIFO (Times before) or FILO (First in first).

Picture 2 appeared first on front

There are many examples of piles in real life. Consider the example of the stacked plates

dit each other in the canteen. Plate at the top is the first plate is removed, the plate is

has been put in place camping fixed on the pile for the longest time.

Therefore, it can only be seen after LIFO (Times before) / FILO (first in last

Out) command. In particular, three basic operations the following is made in the pile:

Press: Add items in the stack. If the heap is full, then say is overflowing.

Pop: remove items from the stack. These items appear in reverse chronological order with them

waste product. If empty pile, it is called conditionally overflow.

Peek or Top: Returns the element focus on the same.

isEmpty: Returns true if stack is empty, in contrast to false.

Operation stack and working mechanisms:

Push (To insert an element into the stack)

The process of bringing an element new data to a stack is called manipulation of the press. Minus points

activities include a number of steps:

Step 1 - Check if stack is full yet.

Step 2 - If the stack is full, please create bugs and escape.

Step 3 - If the stack is not full, let's move on to the empty spot next.

Step 4 - Add data elements to a location stack, where the highest score.

Step 5 - return success.

Buddy Raj Gurung (Buddhist Raj Gurung

Three

A simple algorithm for the operation Push can be deduced as follows:

start push procedure: stack, data

if the stack is full

return null

endif

top ← top + 1

data stack [top] ← data

P2 Determine the operations of a memory stack and how it is used to implement function calls in a computer.

1.definition of memory stack

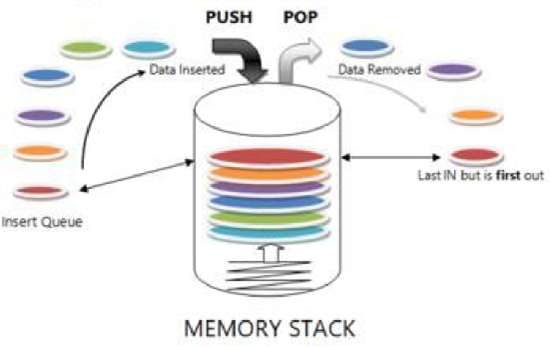
* Stack memory is used for each thread's execution.
* Stack memory includes method-specific values: local variables and references to objects contained in the heap memory referenced by the method.
* Stack memory is referenced in LIFO (Last In First Out) order. That is, stack storage. When a method is executed, a block is created in stack memory to hold local primitives and references to objects. When the method ends, that block will no longer be used and served to the next method.
* Stack memory is very small in size compared to Heap memory.

2. how the memory is organized

Stack is also known as list of Last In First Out (LIFO). It is the most important feature in the CPU. It saves your data so that element is stored last is retrieved first. The stack is a memory unit has a recorded radio address. This register affects the address of the stack, called pointer stack (SP). Pointer stack constant influence to the address of the element located at the top of the stack. It is possible to insert an element into or remove an element from the stack. The activity insert is called operation push and the delete operation is called active turn. In stack machine, the operation is simulated by increasing or decreasing the registers SP. The stack can be arranged as a set of memory words or registers. Consider a 64-word register stack arranged as displayed in the figure. The stack pointer register includes a binary number, which is the address of the element present at the top of the stack. The three-element A, B, and C are located in the stack.

The stack can be arranged as a set of memory words or registers. Consider a 64-word register stack arranged as displayed in the figure. The stack pointer register includes a binary number, which is the address of the element present at the top of the stack. The three-element A, B, and C are located in the stack.

The element C is at the top of the stack and the stack pointer holds the address of C that is 3. The top element is popped from the stack through reading memory word at address 3 and decrementing the stack pointer by 1. Then, B is at the top of the stack and the SP holds the address of B that is 2. It can insert a new word, the stack is pushed by incrementing the stack pointer by 1 and inserting a word in that incremented location.

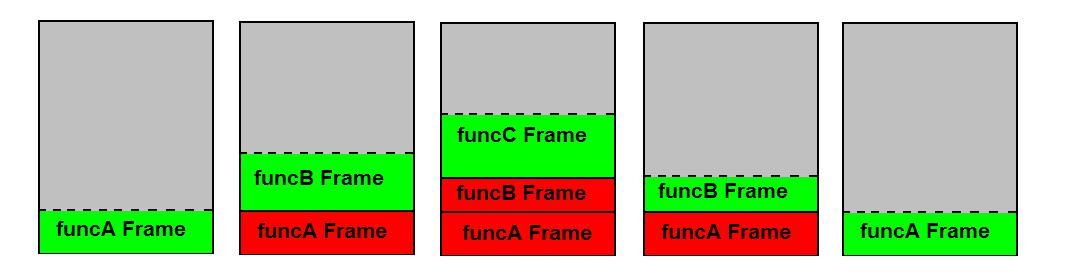


3. how a method (function) calls is implemented with stack

In most implementations, data for a call to function is stored in a record enabled, contains space for each parameter of the function and local variables, objects temporarily, return address, and other items that function needs. In this course, we often only consider the parameters and local variables in a record enabled, ignore the part other data in it.

In memory management, stack based, the record trigger is stored in a data structure called the stack. A stack works just like a stack of pancakes: when a pancake is, it is placed at the top of the paper and when took a pancake off the stack, it is the pancakes on the top is removed. Therefore, the pancakes finally get to do is the is the first feeding, lead to feeding behavior ago, first out (LIFO). The record trigger is similar storage: when a record trigger is created, it will be placed on top of the stack and record the first activation is destroyed is the last record to be created. This leads to a stack frame equivalent term for a record enabled.

When the program is run, the main() function is called, so an activation record is created and added to the top of the stack. Then main() calls foo(), which places an activation record for foo() on the top of the stack. Then bar() is called, so its activation record is put on the stack. When bar() returns, its activation record is removed from the stack. Then foo() completes, removing its activation record. Finally, the activation record for main() is destroyed when the function returns.



P3

1.Definition of software stack

An application consists of a set of functions working together in a defined architecture to deliver specified services to the user. The simplest application architecture consist of three layers:

Presentation Layer - The presentation layer is what the client sees when they access the application through a website or web-based application portal.

Logic Layer - The logic layer contains application logic and business rules that help fulfill application requests. This layer makes calculations and decisions about how to process requests while controlling the transmission of data between the data layer and the presentation layer.

Data Layer - The data layer is a server-side system that passes information to the logic layer when it is necessary to complete a calculation or when it needs to be passed to the presentation layer where it becomes visible to users.

Each of these layers has unique requirements in terms of the programming languages and software tools that are required to establish and maintain its function. A web-based presentation layer may be written in languages like HTML5, Javascript and CSS. The application layer could be programmed in Java, C#, Python or C++. Applications like MySQL and MongoDB could be used to maintain back-end servers.

2. specific abstract data type for software stack

a type definition command of a stack abstract can specify that the status of a stack S can only be modified by the calculations

push (S, x), where x is a number value has properties not determined;

pop (S), the result is to create a value,

with the constraint that

 For any value of x and any variable abstract V how the chain works

{push (S, x); V pop (S)} is equivalent to V x

Because the assignment V ← x, by definition, can not change the status of the S, this condition implies

that V ← pop (S) to restore S about the status that it had before push (S, x). From this condition and from

the properties of the variable abstraction, for example, it follows that the sequence

{push (S, x); push (S, y); U ← pop (S); push (S, z); V ← pop (S); W ← pop (S)}

in which x, y and z is any value and U, V, W are the variables distinguished by pairs, equivalent to

{U ← y; V ← z; W ← x}

Here, we implied that the activity on an individual stack is not modified status

any case ADT else, including the other stacks; that is,

 For all values x, y and any stack to separate S and T, string

{push (S, x); push (T, y)} is equivalent to {push (T, y); push (S, x)}.

A definition stack abstract often also includes a function that has value, Boolean empty (S) and

a math create () returns a stack, with the axiom is equivalent to

create () ≠ S to any stack before it's (a new stack is created differences with all

stack ago);

empty (create ()) (a stack of newly created empty);

not empty (push (S, x)) (to push something into a stack make it not empty)

LAMP (Linux, Apache, MySQL, PHP) - a famous software for web development. The lowest layer of the hierarchy of the stack is the Linux operating system. The highest class of decentralization is scripting language - in this case, PHP programming languages Python or Perl. Stack LAMP popular because of the ingredients all open source and stack can be run on commodity hardware. Unlike the software stack monolithic, often combined tightly and usually built for a specific operating system, stack LAMP is pairing loose. This simply means that even though the component was not originally designed for working together, they have proven to be complementary and are often used together. Today, LAMP the component is included in most Linux distributions.

MEAN (MongoDB, Express, Angular and Node) - a group of development tools known to help to remove the language barriers often encountered in software development. In a stack MEAN, the platform is MongoDB is a database document NoSQL. HTTP server is Express and Angular is a framework for JavaScript front-end. The highest class of the stack is Node, a platform for server side scripting.

Apache CloudStack - a stack cloud management open source used by the big business customer and service provider offering infrastructure as a service (IaaS). CloudStack give the developer more class of service options as well as support for multiple types of supervisors and the program interface of application (API)