RGPV PYQ

Fundamentally, data structures in computer science and programming are important concepts that enable efficient and meaningful organization and manipulation of data.

It offers a way to store and arrange data so that operations such as insertion, deletion, searching and manipulation can be done more efficiently.

There are several types of data structures designed for different types of tasks and requirements.

The following explains some common data structures.

1. Array:

Array is a collection of elements identified by an index or key.

Its elements are kept together in subsequent memory locations with accessing of any element requiring knowledge of its index.

Arrays are useful when you need to retrieve sequential data.

Array example in C language:

C

```c
int numbers[5] = {1, 2, 3, 4, 5};  // Declaration and initialization of an array
int x = numbers[2];                // Accessing the third element
```

```
(index 2)
```

2. Linked list:

The concept of a linked list is a collection of nodes that contain data and links to the next node. Linked lists allow for dynamic memory allocation as well as easy insertion and deletion.

Linked list example in Python:

Python

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

# Creating a linked list
head = Node(1)
head.next = Node(2)
head.next.next = Node(3)
```

3. Stack:

A stack is an example of LIFO data structure. Elements are added and removed from the same end, called the top. Common operations include push (add to the top) and pop (remove from the top).

Stack example in Java language:

Java

```
import java.util.Stack;

Stack<Integer> stack = new Stack<>();
stack.push(1);
stack.push(2);
int x = stack.pop();  // x is 2 (last element added)
```

4. Queue:

A queue refers to a type of FIFO data structure. Elements are added at the rear and removed from the front. Common operations include enqueue (add to the rear) and dequeue (remove from the front).

Queue example in C++ language:

C++

```cpp
#include <queue>

std::queue<int> q;
q.push(1);
q.push(2);
int x = q.front();  // x is 1 (first element added)
q.pop();
```

5. Tree:

A hierarchical tree, has a root node connected by branches. Trees have one parent while others contain children nodes only. Hierarchical structures are modeled by trees all over.

Tree example in Python language:

Python

```python
class TreeNode:
    def __init__(self, value):
        self.value = value
        self.children = []

# Creating a simple tree
root = TreeNode(1)
root.children.append(TreeNode(2))
```

```
root.children.append(TreeNode(3))
```

6. Graph:

Nodes come in pairs in graphs which are simply collections of vertices joined together with edges. A complex relationship can be represented by a graph either undirected or directed.

Graph example in Java language:

Java

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// Creating a simple graph
Map<Integer, List<Integer>> graph = new HashMap<>();
graph.put(1, new ArrayList<>(List.of(2, 3)));
graph.put(2, new ArrayList<>(List.of(1, 4)));
```

Related posts:

1. RGPV BCE PYQs
2. Describe Remote sensing
3. Explain I/O devices in detail with suitable examples

4. Explain memory and type of memory in detail.
5. Define algorithms. What is the need of algorithms ? Describe three benefits of algorithms.
6. Explain procedure-oriented programming with examples.
7. Explain the following: Data Type, Tokens, Variables, Operator
8. Define objects and classes. can a class in C++ have more than one constructor with the same name? Justify your answer with suitable example.