

BIT 2203: DATA STRUCTURES AND ALGORITHMS

Assignment 1

Using a programming language of your choice, in groups, write code to represent each of the data structure classification and types.

Research where the data structures types are applied and give reasons why.

**Give examples of applications that are using the data structure type and algorithm.
Give reasons why.**

Research how data structures and algorithms work within systems.

APPLICATIONS OF DATA STRUCTURES & ALGORITHMS IN SYSTEMS

1. ARRAY

Where Arrays Are Applied

Operating systems, databases, embedded systems, scientific computing, web applications.

Example Applications

Student management systems, image processing software, e-commerce platforms.

Algorithms Used

Linear Search, Binary Search, Bubble Sort, Quick Sort.

Reasons for Using Arrays

Fast access ($O(1)$), simple structure, efficient for fixed-size data.

How Arrays Work Within Systems

Arrays use contiguous memory allocation, allowing fast access and high performance.

The screenshot shows the Falcon C++ IDE interface. The main window displays a C program named 'Newfile.c' with the following code:

```
#include <stdio.h>
int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    printf("First element: %d\n", arr[0]);
    int i;
    for(i = 0; i < 5; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

The code is color-coded for syntax highlighting. The IDE also includes a Projects view, an Outline view showing 'Includes' and 'main(): int', and status bars at the bottom indicating the current file is 'Newfile.c', line 8, column 25, and selected 0 characters.

2. LINKED LIST

Where Applied

Memory management, dynamic storage, system software, multimedia applications.

Example Applications

Music playlists, undo/redo systems, file navigation.

Algorithms Used

Traversal, insertion, deletion.

Reasons

Dynamic size, efficient insertion and deletion.

How They Work

Nodes stored in non-contiguous memory linked by pointers.

The screenshot shows the Falcon C++ IDE interface. The main window displays a C file named 'NewFile.c' with the following code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* next;
7 };
8
9 int main() {
10     struct Node *head, *second, *third;
11
12     head = (struct Node*)malloc(sizeof(struct Node));
13     second = (struct Node*)malloc(sizeof(struct Node));
14     third = (struct Node*)malloc(sizeof(struct Node));
15
16
17     head->data = 10;
18     head->next = second;
19
20     second->data = 20;
21     second->next = third;
22
23     third->data = 30;
24     third->next = NULL;
25
26     struct Node* temp = head;
27     while(temp != NULL) {
28         printf("%d -> ", temp->data);
29         temp = temp->next;
30     }
31     printf("NULL");
32
33
34 }
35
```

The status bar at the bottom indicates "Compilation success in 0.0235 s". The Outline panel on the right shows the project structure with files 'Includes', 'Node', and 'main(): int'.

3. STACK

Where Applied

Operating systems, compilers, browsers.

Example Applications

Browser navigation, function calls, undo/redo.

Algorithms Used

Push, Pop, Peek.

Reasons

LIFO principle, fast operations.

How They Work

Used in call stacks for function execution.

The screenshot shows the Falcon C++ IDE interface. The main window displays a code editor with the file 'NewFile.c' open. The code implements a stack using an array and two functions: push and pop. The push function increments a top pointer and stores the value at the new top location. The pop function prints the element at the current top and then decrements the top pointer. The main function demonstrates this by pushing three integers (10, 20, 30) and then popping them off. The 'Outline' panel on the right shows the project structure and the definitions of the stack, top, push, pop, and main variables and functions.

```
1 #include <stdio.h>
2 #define SIZE 5
3
4 int stack[SIZE];
5 int top = -1;
6
7 void push(int value) {
8     stack[++top] = value;
9 }
10
11 void pop() {
12     printf("Popped element: %d\n", stack[top--]);
13 }
14
15 int main() {
16     push(10);
17     push(20);
18     push(30);
19
20     pop();
21
22     return 0;
23 }
```

4. QUEUE

Where Applied

Operating systems, networking, real-time systems.

Example Applications

Printer spooling, CPU scheduling.

Algorithms Used

Enqueue, Dequeue.

Reasons

FIFO principle, fairness.

How They Work

Manages waiting processes in order.

The screenshot shows the Falcon C++ IDE interface. The main window displays a C source code file named 'NewFile.c'. The code implements a queue using an array 'queue' of size 5. It includes a '#include <stdio.h>' directive and defines constants for SIZE (5) and front/rear indices (-1). The 'enqueue' function adds elements to the rear of the queue, and the 'dequeue' function removes elements from the front. The 'main' function demonstrates this by enqueueing three values (10, 20, 30) and then dequeuing them. The right side of the interface features an 'Outline' panel showing the project structure and symbols defined in the code.

```
1 #include <stdio.h>
2 #define SIZE 5
3
4 int queue[SIZE];
5 int front = -1, rear = -1;
6
7 void enqueue(int value) {
8     if(front == -1)
9         front = 0;
10    queue[++rear] = value;
11 }
12
13 void dequeue() {
14     printf("Dequeued element: %d\n", queue[front++]);
15 }
16
17 int main() {
18     enqueue(10);
19     enqueue(20);
20     enqueue(30);
21
22     dequeue();
23
24     return 0;
25 }
```

5. TREE

Where Applied

Databases, file systems, AI.

Example Applications

Folder structures, database indexing.

Algorithms Used

Tree traversal, searching.

Reasons

Hierarchical representation, fast searching.

How They Work

Balanced trees reduce disk access time.

The screenshot shows the Falcon C++ IDE interface. The main window displays a C file named 'NewFile.c' with the following code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node* left;
7     struct Node* right;
8 };
9
10 struct Node* createNode(int value) {
11     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
12     newNode->data = value;
13     newNode->left = NULL;
14     newNode->right = NULL;
15     return newNode;
16 }
17
18 void preorder(struct Node* root) {
19     if(root != NULL) {
20         printf("%d ", root->data);
21         preorder(root->left);
22         preorder(root->right);
23     }
24 }
25
26 int main() {
27     struct Node* root = createNode(1);
28     root->left = createNode(2);
29     root->right = createNode(3);
30
31     preorder(root);
32     return 0;
33 }
34
```

The status bar at the bottom indicates "Compilation success in 0.0172 s". The Outline panel on the right shows the project structure with files like 'Includes', 'Node', 'createNode(int)', 'preorder(struct Node)', and 'main()'.

6. GRAPH

Where Applied

Networks, social media, navigation.

Example Applications

Google Maps, social networks.

Algorithms Used

BFS, DFS, Dijkstra.

Reasons

Models relationships, efficient routing.

How They Work

Represents interconnected systems.

Falcon C++

File Edit Search View Project Run Tools Help

Projects Newfile.c

```
1 #include <stdio.h>
2
3 int main() {
4     int graph[4][4] = {
5         {0, 1, 1, 0},
6         {1, 0, 0, 1},
7         {1, 0, 0, 0},
8         {0, 1, 0, 0}
9     };
10
11    int i,j;
12    for( i = 0; i < 4; i++) {
13        for( j = 0; j < 4; j++) {
14            printf("%d ", graph[i][j]);
15        }
16        printf("\n");
17    }
18
19    return 0;
20
21}
```

Newfile.c Ln:13 Col:13 Sel:0 Compilation success in 0.0.157 s

Outline

- Includes
- main():int