



CSA03 - DATA STRUCTURES

LIST OF PROGRAMS

1. Write a C program to perform Matrix Multiplication

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named matmul.cpp. The program prompts the user for the dimensions of two matrices and their elements, then performs matrix multiplication and prints the result. The compiler log shows the compilation process, and the status bar at the bottom provides system information like weather and date.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main(){
4     int a[10][10],b[10][10],mul[10][10],r,c,i,j,k;
5     system("cls");
6     printf("enter the number of row=");
7     scanf("%d",&r);
8     printf("enter the number of column=");
9     scanf("%d",&c);
10    printf("enter the first matrix element=\n");
11    for(i=0;i<r;i++)
12    {
13        for(j=0;j<c;j++)
14        {
15            scanf("%d",&a[i][j]);
16        }
17    }
18    printf("enter the second matrix element=\n");
19    for(i=0;i<r;i++)
20    {
21        for(j=0;j<c;j++)
22        {
23            scanf("%d",&b[i][j]);
24        }
25    }
```

Output window:

```
enter the number of row=3
enter the number of column=3
enter the first matrix element=
1 2 3
1 2 3
1 2 3
enter the second matrix element=
3 2 1
3 2 1
3 2 1
multiply of the matrix=
18   12   6
18   12   6
18   12   6
```

Compiler Log:

```
- Executing: 0
- Output Filename: C:\Users\balaji katta\Downloads\matmul.exe
- Output Size: 324.2958984375 Kib
- Compilation Time: 0.50s
```

Status Bar:

```
Line: 49 Col: 3 Sel: 0 Lines: 49 Length: 927 Insert Done parsing in 0.078 seconds
84°F Partly cloudy
ENG IN 19:47
26-09-2022
```

2. Write a C program to find Odd or Even number from a given set of numbers

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named 'even r odd.cpp' with the following content:

```
1 #include <stdio.h>
2 int main() {
3     int num;
4     printf("Enter an integer: ");
5     scanf("%d", &num);
6     if(num % 2 == 0)
7         printf("%d is even.\n", num);
8     else
9         printf("%d is odd.\n", num);
10    printf("k.balaji\n192110707");
11    return 0;
12 }
13 }
```

The output window shows the execution of the program:

```
C:\Users\balaji katta\Downloads\even r odd.exe
Enter an integer: 5
5 is odd.
k.balaji
192110707
-----
Process exited after 0.6924 seconds with return value 0
Press any key to continue . . .
```

The compiler log window at the bottom shows the following compilation details:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\even r odd.exe
- Output Size: 322.7841796875 KiB
- Compilation Time: 0.31s

The system tray at the bottom right indicates the date and time as 26-09-2022 11:47.

3. Write a C program to find Factorial of a given number without using Recursion

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named factorial.cpp. The program prompts the user for a number, calculates its factorial using a for loop, and prints the result along with the user's name and ID. The compiler log shows no errors or warnings, and the output window shows the execution of the program.

```
#include<stdio.h>
int main(){
    int x,fact=1,n;
    printf("Enter a number to find factorial: ");
    scanf("%d",&n);
    for(x=1;x<=n;x++)
        fact=fact*x;
    printf("Factorial of %d is: %d\n",n,fact);
    printf("k.balaji\n192110707");
    return 0;
}
```

Output window:

```
C:\Users\balaji katta\Downloads\factorial.exe
Enter a number to find factorial: 5
Factorial of 5 is: 120
k.balaji
192110707
-----
Process exited after 0.867 seconds with return value 0
Press any key to continue . . .
```

Compiler Log:

```
Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\factorial.exe
- Output Size: 322.7841796875 Kib
- Compilation Time: 0.30s
```

4. Write a C program to find Fibonacci series without using Recursion

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named fib.cpp. The program prompts the user for the number of elements in the series, initializes variables n1, n2, and n3, and then uses a for loop to calculate the series until it reaches the specified number of elements. The compiler log shows no errors or warnings, and the output window shows the execution of the program.

```
#include<stdio.h>
int main()
{
    int n1=0,n2=1,n3,i,number;
    printf("Enter the number of elements:");
    scanf("%d",&number);
    printf("\n%d %d",n1,n2);
    for(i=2;i<number;i++)
    {
        n3=n1+n2;
        printf(" %d",n3);
        n1=n2;
        n2=n3;
    }
    printf("\nk.balaji\n192110707");
    return 0;
}
```

Output window:

```
C:\Users\balaji katta\Downloads\fib.exe
Enter the number of elements:5
0 1 1 2 3
k.balaji
192110707
-----
Process exited after 1.374 seconds with return value 0
Press any key to continue . . .
```

Compiler Log:

```
Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\fib.exe
- Output Size: 322.7841796875 Kib
- Compilation Time: 0.31s
```

5. Write a C program to find Factorial of a given number using Recursion

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named 'factorial using.cpp' which calculates the factorial of a given number using recursion. The terminal window shows the output of the program, which asks for a number, prints its factorial, and then exits. The compiler log shows no errors or warnings.

```
#include<stdio.h>
int fact(int n);
int main()
{
    int x,n;
    printf(" Enter the Number to Find Factorial :");
    scanf("%d",&n);

    x=fact(n);
    printf(" Factorial of %d is %d",n,x);

    return 0;
}
int fact(int n)
{
    if(n==0)
        return(1);
    return(n*fact(n-1));
}
```

Compiler Log:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\fatorial using.exe
- Output Size: 322.8212890625 Kib
- Compilation Time: 0.33s

System Status:

- Line: 20 Col: 2 Sel: 0 Lines: 20 Length: 306 Insert Done parsing in 0.016 seconds
- 84°F Partly cloudy
- ENG IN 19:52 26-09-2022

6. Write a C program to find Fibonacci series using Recursion

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named 'fib rec.cpp' which generates a Fibonacci series using recursion. The terminal window shows the output of the program, which asks for a number and then prints the Fibonacci series up to that number. The compiler log shows no errors or warnings.

```
#include<stdio.h>
int Fibonacci(int);
int main()
{
    int n, i = 0, c;
    scanf("%d",&n);
    printf("Fibonacci series\n");
    for ( c = 1 ; c <= n ; c++ )
    {
        printf("%d\n", Fibonacci(i));
        i++;
    }
    return 0;
}
int Fibonacci(int n)
{
    if ( n == 0 )
        return 0;
    else if ( n == 1 )
        return 1;
    else
        return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

Compiler Log:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\fib rec.exe
- Output Size: 322.984375 Kib
- Compilation Time: 0.31s

System Status:

- Line: 19 Col: 22 Sel: 0 Lines: 23 Length: 404 Insert Done parsing in 0 seconds
- 84°F Partly cloudy
- ENG IN 19:56 26-09-2022

7. Write a C program to implement Array operations such as Insert, Delete and Display.

The screenshot shows the Embarcadero Dev-C++ IDE interface. On the left, the code editor displays `array insert.cpp` with C code for inserting and deleting elements from an array using switch cases. On the right, the terminal window shows the execution of the program, which asks for the size of the array, lists its elements, inserts a new element at index 2, and then deletes the element at index 2, resulting in a modified list. The status bar at the bottom indicates the date and time as 27-09-2022 11:22.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a[10];
    int elements,i,loc,size,n,j,choice;
    printf("C Program to Insert and Delete an Element in an Array using switch case\n");
    printf("1. Inserting an Element in an Array\n");
    printf("2. Deleting an Element in an Array\n");
    printf("3. Displaying all Element in an Array\n");
    printf("Select your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            printf("Enter the size of an array\n");
            scanf("%d",&size);
            printf("Enter 10 array elements\n",size);
            for(i=0;i<size;i++)
            {
                scanf("%d",&a[i]);
            }
            printf("List before Insertion: ");
            for(i=0;i<size;i++)
            {
                printf("%d ",a[i]);
            }
            printf("\nEnter an element to insert\n");
            scanf("%d",&element);
            printf("Enter the position to insert an element %d\n",element);
            scanf("%d",&loc);
            loc--;
            for(i=size-1;i>=loc;i--)
            {
                a[i+1]=a[i];
            }
            a[loc]=element;
            printf("List after Insertion: ");
            for(i=0;i<size;i++)
            {
                printf("%d ",a[i]);
            }
            break;
        case 2:
            printf("Enter the size of an array\n");
            scanf("%d",&size);
            printf("Enter an element to delete\n");
            scanf("%d",&element);
            for(i=0;i<size;i++)
            {
                if(a[i]==element)
                {
                    a[i]=a[i+1];
                }
            }
            printf("List after deletion\n");
            for(i=0;i<size;i++)
            {
                printf("%d ",a[i]);
            }
            break;
    }
    case 3:
        printf("Enter the size of an array\n");
        scanf("%d",&size);
        printf("Enter 10 array elements\n",size);
        for(i=0;i<size;i++)
        {
            scanf("%d",&a[i]);
        }
        printf("List before Insertion: ");
        for(i=0;i<size;i++)
        {
            printf("%d ",a[i]);
        }
        printf("\nEnter an element to insert\n");
        scanf("%d",&element);
        printf("Enter the position to insert an element %d\n",element);
        scanf("%d",&loc);
        loc--;
        for(i=size-1;i>=loc;i--)
        {
            a[i+1]=a[i];
        }
        a[loc]=element;
        printf("List after Insertion: ");
        for(i=0;i<size;i++)
        {
            printf("%d ",a[i]);
        }
        break;
    }
}
```

8. Write a C program to search a number using Linear Search method

The screenshot shows the Embarcadero Dev-C++ IDE interface. On the left, the code editor displays `linear search.c` with C code for linear search. On the right, the terminal window shows the execution of the program, which asks for the number of elements, enters array elements, and then searches for a specific element (8) found at index 5. The status bar at the bottom indicates the date and time as 27-09-2022 11:28.

```
#include <stdio.h>
int main()
{
    int a[20],i,x,n;
    printf("How many elements?");
    scanf("%d",&n);
    printf("Enter array elements:n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("nEnter element to search:");
    scanf("%d",&x);
    for(i=0;i<n;i++)
    {
        if(a[i]==x)
        {
            break;
        }
    }
    if(i==n)
    {
        printf("Element found at index %d",i);
    }
    else
    {
        printf("Element not found");
    }
}
```

9. Write a C program to search a number using Binary Search method

```

C:\Users\balaji katta\Downloads\binarysearch.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Classes binarysearch.cpp
main()
1 #include <stdio.h>
2 int main()
3 {
4     int i, low, high, mid, n, key, array[100];
5     printf("Enter number of elements:");
6     scanf("%d",&n);
7     printf("Enter %d integers:", n);
8     for(i = 0; i < n; i++)
9         scanf("%d",&array[i]);
10    printf("Enter value to find:");
11    scanf("%d", &key);
12    low = 0;
13    high = n - 1;
14    mid = (low+high)/2;
15    while (low <= high) {
16        if(array[mid] < key)
17            low = mid + 1;
18        else if (array[mid] == key) {
19            printf("%d found at location %d.n", key, mid+1);
20            break;
21        }
22        else
23            high = mid - 1;
24            mid = (low + high)/2;
25    }
26    if(low > high)
27        printf("Not found! %d isn't present in the list.n", key);
28    return 0;
29 }

Compiler Resources Compile Log Debug Find Results Console Close
Abort Compilation
Errors: 0 Warnings: 0 Output Filename: C:\Users\balaji katta\Downloads\binarysearch.exe
Output Size: 323.2841796875 Kib Compilation Time: 0.31s
Shorten compiler pat

Line: 9 Col: 23 Sel: 0 Lines: 29 Length: 604 Insert Done parsing in 0 seconds
91°F Mostly cloudy
ENG IN 11:34 27-09-2022

```

10. Write a C program to implement Linked list operations

```

C:\Users\balaji katta\Downloads\linkedlist.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Classes linkedlist.cpp
Node struct
Delete(int x, List l);
Display(List l);
FindPrevious(int x);
Insert(int x, List l, Position p);
isLast(Position p);
main();
Merge(List l, List l1);

1 #include<stdio.h>
2 #include<stdlib.h>
3 struct Node;
4 typedef struct Node * PtrToNode;
5 typedef PtrToNode List;
6 typedef PtrToNode Position;
7
8 struct Node
9 {
10     int e;
11     Position next;
12 };
13
14 void Insert(int x, List l, Position p)
15 {
16     Position TmpCell;
17     TmpCell = (struct Node*) malloc(sizeof(struct Node));
18     if(TmpCell == NULL)
19         printf("Memory out of space\n");
20     else
21     {
22         TmpCell->e = x;
23         TmpCell->next = p->next;
24         p->next = TmpCell;
25     }
26 }
27
28 int isLast(Position p)
29 {
30     if(p->next == NULL)
31         return 1;
32     else
33         return 0;
34 }

Compiler Resources Compile Log Debug Find Results Console Close
Abort Compilation
Errors: 0 Warnings: 0 Output Filename: C:\Users\balaji katta\Downloads\linkedlist.exe
Output Size: 325.16796875 Kib Compilation Time: 0.31s
Shorten compiler pat

C:\Users\balaji katta\Downloads\linkedlist.exe
Enter the choice :: 1
Enter the element to be inserted :: 1
Enter the position of the element :: 1
1. INSERT 2. DELETE 3. MERGE 4. PRINT 5. QUIT
Enter the choice :: 1
Enter the element to be inserted :: 3
Enter the position of the element :: 2
1. INSERT 2. DELETE 3. MERGE 4. PRINT 5. QUIT
Enter the choice :: 1
Enter the element to be inserted :: 4
Enter the position of the element :: 2
1. INSERT 2. DELETE 3. MERGE 4. PRINT 5. QUIT
Enter the choice :: 4
The list element are :: 1 -> 4 -> 3 ->
1. INSERT 2. DELETE 3. MERGE 4. PRINT 5. QUIT
Enter the choice ::

Line: 143 Col: 2 Sel: 0 Lines: 143 Length: 2930 Insert Done parsing in 0.031 seconds
91°F Mostly cloudy
ENG IN 11:38 27-09-2022

```

11. Write a C program to implement Stack operations such as PUSH, POP and PEEK

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named `push pop.cpp`. The program defines a stack of integers and implements basic operations: Push, Pop, Show, and Exit. The stack is represented by an array `inp_array` of size 4. The `main` function provides a menu for these operations and handles user input. The output window shows the execution of the program, including the menu options and the resulting stack elements.

```
#include<stdio.h>
#include<stdlib.h>
#define Size 4
int Top=-1, inp_array[Size];
void Push();
void Pop();
void show();
int main()
{
    int choice;
    while(1)
    {
        printf("\nOperations performed by Stack");
        printf("\n1.Push the element\n2.Pop the element\n3.Show\n4.End");
        printf("\nEnter the choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: Push();
            break;
            case 2: Pop();
            break;
            case 3: show();
            break;
            case 4: exit(0);
            break;
        }
    }
}
```

Output window:

```
C:\Users\balaji katta\Downloads\push pop.exe
3.Show
4.End

Enter the choice:1
Enter element to be inserted to the stack:
3

Operations performed by Stack
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice:3
Elements present in the stack:
3
3
2

Operations performed by Stack
1.Push the element
2.Pop the element
3.Show
4.End

Enter the choice:
```

Compiler Log:

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\push pop.exe
- Output Size: 324.05078125 Kib
- Compilation Time: 0.42s
```

12. Write a C program to implement the application of Stack (Notations)

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named `stack.cpp`. This program implements a stack of characters and performs arithmetic expression evaluation based on operator precedence. It uses a priority function to determine the precedence of operators. The `main` function reads an expression from the user and evaluates it using the stack. The output window shows the execution of the program, including the expression and the result.

```
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top = -1;
void push(char x)
{
    stack[++top] = x;
}
char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}
int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}
int main()
{
    char exp[100];
    char e, x;
```

Output window:

```
C:\Users\balaji katta\Downloads\stack.exe
Enter the expression : a+b*(c-d/e)*g
a b c d e f / g * + -
Process exited after 24.88 seconds with return value 0
Press any key to continue . . .
```

Compiler Log:

```
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Balaji katta\Downloads\stack.exe
- Output Size: 324.225515625 Kib
- Compilation Time: 0.31s
```

13. Write a C program to implement Queue operations such as ENQUEUE, DEQUEUE and Display

```

C:\Users\balaji katta\Downloads\queue.c - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Classes [*] push pop.cpp [*] stack.cpp [*] queue.c *
1 #include <stdio.h>
2 # define SIZE 100
3 void enqueue();
4 void dequeue();
5 void show();
6 int inp_arr[SIZE];
7 int Rear = - 1;
8 int Front = - 1;
9 main()
10 {
11     int ch;
12     while (1)
13     {
14         printf("1.Enqueue Operation\n");
15         printf("2.Dequeue Operation\n");
16         printf("3.Display the Queue\n");
17         printf("4.Exit\n");
18         printf("Enter your choice of operations : ");
19         scanf("%d", &ch);
20         switch (ch)
21         {
22             case 1:
23                 enqueue();
24                 break;
25             case 2:
26                 dequeue();
27                 break;
28             case 3:
29                 show();
30         }
31     }
32 }

```

C:\Users\balaji katta\Downloads\queue.exe

- 1.Enqueue Operation
- 2.Dequeue Operation
- 3.Display the Queue
- 4.Exit

Enter your choice of operations : 1

Element to be inserted in the Queue : 20

- 1.Enqueue Operation
- 2.Dequeue Operation
- 3.Display the Queue
- 4.Exit

Enter your choice of operations : 1

Element to be inserted in the Queue : 21

- 1.Enqueue Operation
- 2.Dequeue Operation
- 3.Display the Queue
- 4.Exit

Enter your choice of operations : 3

Queue:

- 1.Enqueue Operation
- 2.Dequeue Operation
- 3.Display the Queue
- 4.Exit

Enter your choice of operations :

14. Write a C program to implement the Tree Traversals (Inorder, Preorder, Postorder)

```

C:\Users\balaji katta\Downloads\tree traversal.cpp - [Executing] - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Project Classes [*] arrayinsertion.cpp [*] array insert.cpp [*] tree traversal.cpp [*] hashing using linear probing.cpp [*] mergesorting.cpp [*] quicksort.cpp *
Line: 13 Col: 6 Sel: 0 Lines: 83 Length: 1624 Insert Done parsing in 0.016 seconds
Message
9 1 C:\Users\balaji katta\Downloads\tree traversal.c [Warning] return type defaults to 'int' [-Wimplicit-int]
Cloudy 31°C
ENG IN 11:26 28-09-2022

```

```

node struct
main() int
newNode(int data)
printInorder(struct)
printPostorder(struct)
printPreorder(struct)

struct node* newNode(int data)
{
    struct node* node
    node = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

void printPostorder(struct node* node)
{
    if (node == NULL)
        return;

    printPostorder(node->left);

    printPostorder(node->right);
}

```

C:\Users\balaji katta\Downloads\tree traversal.exe

Preorder traversal of binary tree is

- 1
- 2
- 4
- 5
- 3

Inorder traversal of binary tree is

- 4
- 2
- 5
- 1
- 3

Postorder traversal of binary tree is

- 4
- 5
- 2
- 3
- 1

Compiler (1) Resources Compile Log Debug Find Results Console Close

Abort Compilation

Shorten compiler pat

Errors: 0 Warnings: 0 Output Filename: C:\Users\balaji katta\Downloads\tree traversal.exe Output Size: 324.107421075 Kib Compilation Time: 0.34s

Cloudy 91°F

15. Write a C program to implement hashing using Linear Probing method

The screenshot shows the Embarcadero Dev-C++ IDE interface. The project navigation bar at the top lists several files: arrayinsrtion.cpp, array insert.cpp, tree traversal.cpp, hashing using linear probing.cpp, and others. The main code editor window displays a C program for linear probing hashing. The code includes declarations for display, insert, search functions, and a hash table h of size TABLE_SIZE. It prompts the user to enter a value to insert into the hash table, handles collisions using linear probing, and prints the contents of the hash table. The compiler output window at the bottom shows no errors or warnings, and the system tray indicates it's 91°F cloudy.

```
#include <stdio.h>
#include<stdlib.h>
#define TABLE_SIZE 10
int h[TABLE_SIZE]={NULL};
void insert()
{
    int key,index,i,flag=0,hkey;
    printf("enter a value to insert into hash table\n");
    scanf("%d",&key);
    hkey=key%TABLE_SIZE;
    for(i=0;i<TABLE_SIZE;i++)
    {
        index=(hkey+i)%TABLE_SIZE;
        if(h[index] == NULL)
        {
            h[index]=key;
            break;
        }
    }
    if(i == TABLE_SIZE)
        printf("\nelement cannot be inserted\n");
}
void display()
{
    for(i=0;i<TABLE_SIZE;i++)
    {
        if(h[i] != NULL)
            printf("at index %d value = %d\n",i,h[i]);
    }
}
void search()
{
    int key,value;
    printf("enter search element");
    scanf("%d",&key);
    value=h[key%TABLE_SIZE];
    if(value==NULL)
        printf("value is not found at index %d",key%TABLE_SIZE);
    else
        printf("value is found at index %d",key%TABLE_SIZE);
}
```

Compiler (1) Resources Compile Log Debug Find Results Console Close

Line: 1 Col: 1 Sel: 0 Lines: 83 Length: 1479 Insert Done parsing in 0.015 seconds

Cloudy 91°F ENG IN 13:23 28-09-2022

16. Write a C program to arrange a series of numbers using Insertion Sort

The screenshot shows the Embarcadero Dev-C++ IDE interface. The project navigation bar at the top lists several files: arrayinsrtion.cpp, array insert.cpp, tree traversal.cpp, hashing using linear probing.cpp, mergesorting.cpp, quicksort.cpp, and heap.cpp. The main code editor window displays a C program for insertion sort. It asks the user for the number of elements and then for the elements themselves. It then performs insertion sort on the array and prints the sorted elements. The compiler output window at the bottom shows no errors or warnings, and the system tray indicates it's 92°F cloudy.

```
#include <stdio.h>
#include<conio.h>
int main()
{
    int a[10],i,j,k,n;
    printf("How many elements you want to sort?\n");
    scanf("%d",&n);
    printf("\nEnter the Elements into an array:\n");
    for (i=0;i<n;i++)
        scanf("%d",&a[i]);
    for(i=1;i<n;i++)
    {
        k=a[i];
        for(j=i-1; j>=0 && k<a[j]; j--)
            a[j+1]=a[j];
        a[j+1]=k;
    }
    printf("\n Elements after sorting: \n");
    for(i=0;i<n;i++)
        printf("%d\n", a[i]);
    getch();
}
```

Compiler (1) Resources Compile Log Debug Find Results Console Close

Line: 21 Col: 2 Sel: 0 Lines: 21 Length: 434 Insert Done parsing in 0.016 seconds

Cloudy 92°F ENG IN 13:33 28-09-2022

17. Write a C program to arrange a series of numbers using Merge Sort

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named 'mergsorting.cpp' which implements the Merge Sort algorithm. The main function initializes an array 'a' with values {10, 14, 19, 26, 27, 31, 33, 35, 42, 44} and merges it into another array 'b'. The compiler log shows no errors or warnings, and the output window shows the list before and after sorting.

```
#include <stdio.h>
#define max 10
int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
int b[10];
void merging(int low, int mid, int high) {
    int l1, l2, i;
    for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
        if(a[l1] <= a[l2])
            b[i] = a[l1++];
        else
            b[i] = a[l2++];
    }
    while(l1 <= mid)
        b[i++] = a[l1++];
    while(l2 <= high)
        b[i++] = a[l2++];
    for(i = low; i <= high; i++)
        a[i] = b[i];
}
void sort(int low, int high) {
    int mid;
    if(low < high) {
        int mid;
        sort(low, mid);
        sort(mid+1, high);
        merging(low, mid, high);
    }
}
int main() {
    sort(0, max-1);
    return 0;
}
```

Compiler Log:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\mergsorting.exe
- Output Size: 323.376953125 Kib
- Compilation Time: 0.30s

Output Window:

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
Process exited after 0.02756 seconds with return value 0
Press any key to continue . . .
```

18. Write a C program to arrange a series of numbers using Quick Sort

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program named 'quicksort.cpp' which implements the Quick Sort algorithm. The main function prompts the user for the number of elements and then sorts an array 'number' of size 25. The compiler log shows no errors or warnings, and the output window shows the user input and the sorted array.

```
#include<stdio.h>
void quicksort(int number[25],int first,int last){
    int i, j, pivot, temp;
    if(first>last){
        pivot=first;
        i=first;
        j=last;
        while(i<j){
            while(number[i]<=number[pivot]&&i<last)
                i++;
            while(number[j]>number[pivot])
                j--;
            if(i<j){
                temp=number[i];
                number[i]=number[j];
                number[j]=temp;
            }
        }
        temp=number[pivot];
        number[pivot]=number[j];
        number[j]=temp;
        quicksort(number,first,j-1);
        quicksort(number,j+1,last);
    }
}
int main(){
    int i, count, number[25];
    printf("How many elements are u going to enter?: ");
    scanf("%d",&count);
    for(i=0;i<count;i++)
        number[i]=rand()%100;
}
```

Compiler Log:

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\balaji katta\Downloads\quicksort.exe
- Output Size: 323.318359375 Kib
- Compilation Time: 0.31s

Output Window:

```
How many elements are u going to enter?: 5
Enter 5 elements: 9 8 7 6 2
Order of Sorted elements: 2 6 7 8 9
Process exited after 4.747 seconds with return value 0
Press any key to continue . . .
```

19. Write a C program to implement Heap sort.

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a file named 'heap.cpp' containing C code for heap sort. The code includes functions for heapify, heapSort, and printing array elements. The compiler log shows successful compilation with no errors or warnings. The output window shows the execution of the program, which sorts an array of integers from 48 to 1. The taskbar at the bottom indicates the system is running at 91°F Cloudy, ENG IN, and the date is 28-09-2022.

```
#include <stdio.h>
void heapify(int a[], int n, int i)
{
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && a[left] > a[largest])
        largest = left;
    if (right < n && a[right] > a[largest])
        largest = right;
    if (largest != i)
    {
        int temp = a[i];
        a[i] = a[largest];
        a[largest] = temp;
        heapify(a, n, largest);
    }
}
void heapSort(int a[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(a, n, i);
    for (int i = n - 1; i >= 0; i--)
    {
        int temp = a[0];
        a[0] = a[i];
        a[i] = temp;
        heapify(a, i, 0);
    }
}
```

Before sorting array elements are -
48 10 23 43 28 26 1
After sorting array elements are -
1 10 23 26 28 43 48
Process exited after 0.03005 seconds with return value 0
Press any key to continue . . .

Compiler (1) Resources Compile Log Debug Find Results Console Close

Line: 52 Col: 2 Sel: 0 Lines: 52 Length: 1256 Insert Done parsing in 0.016 seconds

91°F Cloudy ENG IN 13:30 28-09-2022

20. Write a program to perform the following operations:

- Insert an element into a AVL tree.
- Delete an element from a AVL tree.
- Search for a key element in a AVL tree.

The screenshot shows the Embarcadero Dev-C++ IDE interface. The project navigation bar at the top lists several files: arryinsertion.cpp, array insert.cpp, tree traversal.cpp, hashing using linear probing.cpp, mergesorting.cpp, quicksort.cpp, avltree.cpp, and heap.cpp. The main code editor window displays the `avltree.cpp` file, which contains C++ code for an AVL tree. The code includes structures for nodes, insertion logic, and rotation functions. To the right of the editor is a terminal window showing the execution of the program. The terminal output shows the initial state of the tree (4 2 1 3 7 5 8), the result after deletion (After deletion: 4 2 1 7 5 8), and the final message (Process exited after 0.02864 seconds with return value 0). A message at the bottom of the terminal says "Press any key to continue . . ." Below the editor and terminal are the compiler log and resources windows, which show compilation errors and warnings. The system tray at the bottom right indicates the date as 28-09-2022 and the time as 13:39.

21. Write a C program to Graph traversal using Breadth First Search

The screenshot shows the Embarcadero Dev-C++ IDE interface. The project navigation bar at the top lists several files: primes.cpp and bfs1.cpp. The main code editor window displays the `bfs1.cpp` file, which contains C code for Breadth First Search (BFS). The code includes functions for initializing a queue, performing BFS, and printing the results. To the right of the editor is a terminal window showing the execution of the program. The terminal prompts the user for the number of vertices (Enter the number of vertices: 3), the graph data in matrix form (Enter graph data in matrix form: 6 5 8 4 5 6 2 3 1), and the starting vertex (Enter the starting vertex: 1). The output shows the reachable nodes (The node which are reachable are: 1 2 3) and the final message (Process exited after 16.25 seconds with return value 0). A message at the bottom of the terminal says "Press any key to continue . . ." Below the editor and terminal are the compiler log and resources windows, which show compilation errors and warnings. The system tray at the bottom right indicates the date as 29-09-2022 and the time as 14:33.

22. Write a C program to Graph traversal using Depth First Search

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program for Depth First Search (DFS). The main function prompts for the number of vertices and graph data in matrix form, then starts a breadth-first search (BFS) from a specified vertex to find all reachable nodes.

```
#include<stdio.h>
#include<conio.h>
int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;
void bfs(int v) {
    for(i=1;i<=n;i++)
        if(a[v][i] && !visited[i])
            q[++r]=i;
    if(f<r) {
        visited[q[f]]=1;
        bfs(q[f+1]);
    }
}
int main() {
    int v;
    printf("\nEnter the number of vertices:");
    scanf("%d",&n);
    for(i=1;i<=n;i++) {
        q[i]=0;
        visited[i]=0;
    }
    printf("\nEnter graph data in matrix form:\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            scanf("%d",&a[i][j]);
    printf("\nEnter the starting vertex:");
    scanf("%d",&v);
    bfs(v);
    printf("\nThe node which are reachable are:\n");
    for(i=1;i<=n;i++)
        if(visited[i]==1)
            printf("%d ",i);
}
Compiler (1) Resources Compile Log Debug Find Results Console Close
Line: 1 Col: 1 Sel: 0 Lines: 34 Length: 735 Insert Done parsing in 0.015 seconds
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
C:\Users\balaji katta\Downloads\dfs.exe
Enter the number of vertices:3
Enter graph data in matrix form:
2 5 6
2 3 4
9 8 7
Enter the starting vertex:2
The node which are reachable are:
1 2 3
```

The terminal window shows the output of the program, indicating that vertices 1, 2, and 3 are reachable from vertex 2. The compiler log shows no errors or warnings.

23. Implementation of Shortest Path Algorithms using Dijkstra's Algorithm

The screenshot shows the Embarcadero Dev-C++ IDE interface. The code editor displays a C program for Dijkstra's shortest path algorithm. It includes functions for calculating minimum distance, printing the solution, and performing the Dijkstra algorithm itself. The terminal window shows the vertex distances from the source vertex.

```
#include <limits.h>
#include <stdio.h>
#define V 9
int minDistance(int dist[], bool sptSet[]) {
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; ++v)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    return min_index;
}
int printSolution(int dist[], int n) {
    printf("Vertex Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t %d\n", i, dist[i]);
}
void dijkstra(int graph[V][V], int src) {
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;
    dist[src] = 0;
    for (int count = 0; count < V - 1; count++) {
        int u = minDistance(dist, sptSet);
        sptSet[u] = true;
        for (int v = 0; v < V; v++)
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
                printSolution(dist, V);
    }
}
Compiler (2) Resources Compile Log Debug Find Results Console Close
Line: 1 Col: 1 File: C:\Users\balaji katta\Downloads\dje.cpp Message: In function 'int printSolution(int*, int*)':
15 1 C:\Users\balaji katta\Downloads\dje.cpp [Warning] no return statement in function returning non-void [-Wreturn-type]
File: C:\Users\balaji katta\Downloads\dje.cpp
Message: In function 'int minDistance(int*, bool*)':
15 1 C:\Users\balaji katta\Downloads\dje.cpp [Warning] no return statement in function returning non-void [-Wreturn-type]
File: C:\Users\balaji katta\Downloads\dje.cpp
Message: In function 'void dijkstra(int**, int)':
15 1 C:\Users\balaji katta\Downloads\dje.cpp [Warning] no return statement in function returning non-void [-Wreturn-type]
```

The terminal window shows the vertex distances from the source vertex 0. The compiler log shows three warning messages related to the printSolution, minDistance, and dijkstra functions regarding missing return statements.

24. Implementation of Minimum Spanning Tree using Prim's Algorithm

The screenshot shows the Embarcadero Dev-C++ IDE interface. The main window displays the source code for 'primes.cpp' which contains the implementation of Prim's algorithm. The code includes functions for reading input, finding minimum keys, and printing the results. A terminal window shows the execution output, displaying the edges and weights of the Minimum Spanning Tree. The status bar at the bottom provides system information like date and time.

```
#include <stdio.h>
#include <limits.h>
#define V 5
int minKey(int key[], int mstSet[]) {
    int min = INT_MAX, min_index;
    int v;
    for (v = 0; v < V; v++)
        if (mstSet[v] == 0 && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}
int printMST(int parent[], int n, int graph[V][V]) {
    int i;
    printf("Edge  Weight\n");
    for (i = 1; i < V; i++) {
        printf("%d - %d  %d \n", parent[i], i, graph[1][i]);
    }
}
void primMST(int graph[V][V]) {
    int parent[V];
    int key[V], i, v, count;
    int mstSet[V];
    for (i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = 0;
    key[0] = 0;
    parent[0] = -1;
    for (count = 0; count < V - 1; count++) {
        int u = minKey(key, mstSet);
        mstSet[u] = 1;
        for (v = 0; v < V; v++)
            if (graph[u][v] > 0 && mstSet[v] == 0 && graph[u][v] < key[v])
                key[v] = graph[u][v];
    }
}
```

Output from the terminal window:

```
Edge  Weight
0 - 1  2
1 - 2  3
0 - 3  6
1 - 4  5
-----
Process exited after 0.01832 seconds with return value 0
Press any key to continue . . .
```

Compiler Log:

```
C:\Users\balaji katta\Downloads\primes.cpp: In function 'int printMST(int*, int, int (*)[S]):'
C:\Users\balaji katta\Downloads\primes.cpp: [Warning] no return statement in function returning non-void [-Wreturn-type]
```

Status Bar:

```
Line: 40 Col: 2 Sel: 0 Lines: 40 Length: 1198 Insert Done parsing in 0.031 seconds
90°F Haze
```

```
ENG IN 14:32 29-09-2022
```

25. Implementation of Minimum Spanning Tree using Kruskal Algorithm

The screenshot shows the Embarcadero Dev-C++ IDE interface. The project 'kurskal' is open, containing files like primes.cpp, bfs1.cpp, dfs1.cpp, dfs.cpp, dje.cpp, and kurskal.cpp. The 'kurskal.cpp' file is the active code editor, displaying the C++ implementation of Kruskal's algorithm. The code includes comments explaining the steps: finding vertices, reading the number of vertices, reading the cost adjacency matrix, and printing the edges of the Minimum Cost Spanning Tree. The output window shows the execution of the program, prompting for input and displaying the resulting Minimum Cost Spanning Tree edges and minimum cost. The status bar at the bottom provides compiler statistics and system information.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,min;
int mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
int main()
{
    printf("\n\tImplementation of Kruskal's algorithm\n");
    printf("Enter the no. of vertices:");
    scanf("%d",&n);
    printf("\nEnter the cost adjacency matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    printf("The edges of Minimum Cost Spanning Tree are\n");
    while(ne < n)
    {
        for(i=1,min=999;i<=n;i++)
        {
            for(j=1;j <= n;j++)
            {
                if(cost[i][j]<min)
                {
                    min=cost[i][j];
                    u=i;
                    v=j;
                }
            }
        }
        printf("Edge (%d,%d) = %d\n",u,v,min);
        parent[u]=v;
        ne++;
        min=999;
    }
}
```

Implementation of Kruskal's algorithm

Enter the no. of vertices:

Enter the cost adjacency matrix:

1 2 3
4 5 6
7 8 9

The edges of Minimum Cost Spanning Tree are

1 edge (1,2) =2
2 edge (1,3) =3

Minimum cost = 5

Process exited after 19.74 seconds with return value 0

Press any key to continue . . .

Compiler (1) Resources Compile Log Debug Find Results Console Close

Abort Compilation Shorten compiler path

- Errors: 0 - Warnings: 0 - Output Filename: C:\Users\balaji katta\Downloads\kurskal.exe - Output Size: 324.38671875 KiB - Compilation Time: 0.25s

Line: 64 Col: 2 Sel: 0 Lines: 64 Length: 1088 Insert Done parsing in 0.031 seconds

90°F Haze ENG IN 14:38 29-09-2022