~\Desktop\Data Stuctures.cpp

```cpp
//عمرو حمزه عمرو محمد عظيمي
#include<iostream>
#include <cstdlib>
using namespace std;
int main();
void Dynamic_insert_type();
void static_insert_type();
void dynamicMenu();
void myStack();
void stack_print();
void stack_pop ();
int stack_push (int val);
int stack_isempty();
void myQueue();
void queuePrint();
void queuePop();
void queuePush(int x);
int queueIsempty();
void myList();
void insertAtBeginning(int value);
void insertAtEnd(int value);
void deleteValue(int value);
void deleteFirstNode();
void deleteLastNode();
void insertBeforeNode(int beforeValue, int newValue);
void insertAfterNode(int afterValue, int newValue);
void updateNode(int oldVal, int newVal);
void sortList();
void displayList();


///////////////////////////////////////////////////////////////////
struct stack1 {
    int data;
    stack1* last;
};
stack1 *top = NULL , *newTop=NULL;

struct queue1 {
    int data;
    queue1* next;
};
queue1 *queueHead=NULL ,*queueTail=NULL, *newQueue=NULL;

struct Node {
    int data;
    Node* next;
};
Node* head=NULL , *newNode=NULL;

int insert_type_val = 0;
int static_insert_type_val = 0;
////////////////////////////////////////////////////////////////////

void dynamicMenu()
{
    system("cls");
    int select;
        cout << "\n";
    cout << "             Enter The Type Of Structures \n\n";
    cout << "             ----------------------------\n";
    cout << "            |    (Dynamic Structures)    |\n";
    cout << "             ----------------------------\n";
    cout << "            | 1: Stack                   |\n";
    cout << "            | 2: Queue                   |\n";
    cout << "            | 3: List                    |\n";
    cout << "             ----------------------------\n\n";
    cout << "Press 0 to go back\n\n";
    cout << "Enter The Structures Number -> ";
    cin>>select;
    switch(select)
    {
        case 1:
            insert_type_val = 1;
            Dynamic_insert_type();
            break;
        case 2:
            insert_type_val = 2;
            Dynamic_insert_type();
            break;
        case 3:
            insert_type_val = 3;
            Dynamic_insert_type();
            break;
        case 0:
            main();
            break;
        default:
            dynamicMenu();
            break;
    }

}

void Dynamic_insert_type()
{



    system("cls");
    int select;
        cout << "\n";
    cout << "             ----------------------------\n";
    cout << "            |          (Choose)          |\n";
    cout << "             ----------------------------\n";
    cout << "            | 1: Manual Enter            |\n";
    cout << "            | 2: Random                  |\n";
    cout << "             ----------------------------\n\n";
    cout << "Press 0 to go back\n\n";
    cout << "Enter The Number -> ";
    cin>>select;
    switch(select)
    {
        case 1:
            if(insert_type_val == 1)
                myStack();
            else if(insert_type_val == 2)
```

```cpp
                myQueue();
            else if(insert_type_val == 3)
                myList();
        case 2:
            if(insert_type_val == 1)
            {
                for (int i = 0; i < 7; i++)
                {
                    stack_push(rand()%10+1);
                }
            myStack();
            }

            if(insert_type_val == 2)
            {
                for (int i = 0; i < 7; i++)
                {
                    queuePush(rand()%10+1);
                }
            myQueue();
            }

            if(insert_type_val == 3)
            {
                for (int i = 0; i < 7; i++)
                {
                    insertAtEnd(rand()%10+1);
                }
            myList();
            }
            break;
        case 0:
            system("cls");
            dynamicMenu();
            break;
        default:
            system("cls");
            Dynamic_insert_type();
            break;
    }
}
/////////////////////////Stack Start////////////////////////////////

void myStack()
{
    system("cls");
    int op , val;
    cout << "       ---Stack---\n\n";
    cout << "     Content Of Stack\n\n";
    stack_print();
    cout << "*************************\n";
    cout << "The Operation Of Stack\n\n";
    cout << "  1->Push\n";
    cout << "  2->Pop\n\n";
    cout << "Press 0 to go back\n\n";
    cout<<"Insert The Operation Number -> ";
    cin >> op;

    switch(op) {
        case 1:
            cout<<"\nInsert The Value : ";
            cin>> val;
            stack_push(val);
            cout<<"\n";
            myStack();
            break;

        case 2:
            stack_pop();
            cout<<"\n";
            myStack();
            break;

        case 0:
            dynamicMenu();
            break;
        default:
            cout << "Incorrect Operation\n\n";
            system("pause");
            myStack();
            break;
    }
}

int stack_isempty()
{
    if(top == NULL)
        return 0;
    else
        return 1;
}
int stack_push (int val)
{
    newTop=new stack1;
    newTop->data = val;
    newTop->last = NULL;

    if(top != NULL)
        newTop->last = top;

    top = newTop;

}
void stack_pop ()
{
    if(stack_isempty())
    {
        top = newTop->last;
        cout<<"\n";
        cout<<"Delete-> "<<newTop->data<<"\n"<< endl;
        delete newTop;
        newTop = top;
    }
    else
    {
        cout<<"\n";
        cout<<"Stack Is Empty\n\n";
    }
    system("pause");

```

```cpp
}
void stack_print()
{
        if (!stack_isempty())
    {
        cout << "\n      Stack is Empty\n\n";
    }

    stack1 *temp = top;
    while (temp != NULL)
    {
        cout <<"            "<< temp->data << " \n";
        temp = temp->last;
    }
        cout << "\n";

}
//////////////////////////Stack End/////////////////////////////////


////////////////////////Queue Start////////////////////////////////
void myQueue()
{
    int op, val;
    system("cls");
    cout << "       ---Queue---\n\n";
    cout << "     Content Of Queue\n\n";
    queuePrint();
    cout << "**********************\n";
    cout << "The Operation Of Queue\n\n";
    cout << "  1->Push\n";
    cout << "  2->Pop\n\n";
    cout << "Press 0 to go back\n\n";
    cout<<"Insert The Operation Number -> ";
    cin >> op;

    switch(op)
    {
        case 1:
            cout << "\nEnter the Value: ";
            cin >> val;
            queuePush(val);
            cout<<"\n";
            myQueue();
            break;
        case 2:
            queuePop();
            cout<<"\n";
            myQueue();
            break;
        case 0:
            dynamicMenu();
            break;
        default:
            cout << "Incorrect Operation\n";
            system("pause");
            myQueue();
            break;
    }
}
```

```cpp
int queueIsempty()
{
    if (queueHead == NULL)
        return 0;
    else
        return 1;
}
void queuePush(int x)
{
    newQueue = new queue1;
    newQueue->data = x;
    newQueue->next = NULL;
    if (queueHead == NULL)
        queueHead = queueTail = newQueue;
    else
    {
        queueTail->next = newQueue;
        queueTail = newQueue;
    }
}
void queuePop()
{
    if (queueIsempty())
    {
        queue1 *p = queueHead;
        cout <<"\nDelete-> "<<queueHead->data << endl<< endl;
        queueHead = queueHead->next;
        delete p;
    }
    else
        cout << "\nQueue Is Empty\n\n";
        system("pause");
}
void queuePrint()
{
        if (!queueIsempty())
    {
        cout << "\n      Queue Is Empty\n\n";
    }
    queue1 *temp = queueHead;
    while (temp != NULL)
    {
        cout <<" "<< temp->data << " ";
        temp = temp->next;
    }
    cout << "\n\n";
}
///////////////////////////Queue End/////////////////////////////////


////////////////////////////List Start///////////////////////////////
void myList()
{
    int op, val, oldVal, newVal, refVal;
    system("cls");
    cout << "        ---List---\n\n";
    cout << "      Content Of List\n\n";
    displayList();
    cout << "********************\n";
    cout << "The Operation Of List\n\n";
    cout << "  1->Insert value\n";
    cout << "  2->Insert at Beginning\n";
    cout << "  3->Insert at End\n";
    cout << "  4->Insert Before Node\n";
    cout << "  5->Insert After Node\n";
    cout << "  6->Delete value from any\n";
    cout << "  7->Delete the first value\n";
    cout << "  8->Delete the Last value\n";
    cout << "  9->Update Node\n";
    cout << "  10->Sort List\n\n";
    cout << "Press 0 to go back\n\n";
    cout<<"Insert The Operation Number -> ";
    cin >> op;

    switch (op) {
    case 1:
        cout << "\nEnter the value-> ";
        cin >> val;
        insertAtEnd(val);          //Insert value
        myList();
        break;
    case 2:
        cout << "\nEnter the value-> ";
        cin >> val;
        insertAtBeginning(val);      //Insert at Beginning
        myList();
        break;
    case 3:
        cout << "\nEnter the value-> ";
        cin >> val;
        insertAtEnd(val);          //Insert at End
        myList();
        break;
    case 4:
        cout << "\nEnter the Node value-> ";
        cin >> refVal;
        cout << "\nEnter the value to insert: ";
        cin >> val;
        insertBeforeNode(refVal, val);         //Insert Before Node
        myList();
        break;
    case 5:
        cout << "\nEnter the Node value-> ";
        cin >> refVal;
        cout << "\nEnter the value to insert-> ";
        cin >> val;
        insertAfterNode(refVal, val);          //Insert After Node
        myList();
        break;
    case 6:
        cout << "\nEnter the value to delete-> ";
        cin >> val;
        deleteValue(val);        //Delete value from any
        myList();
        break;
    case 7:
        deleteFirstNode();       //Delete the first value
        myList();
        break;
    case 8:
        deleteLastNode();       //Delete the Last value
        myList();
        break;
    case 9:
        cout << "\nEnter the old value-> ";
        cin >> oldVal;
        cout << "\nEnter the new value-> ";
        cin >> newVal;
        updateNode(oldVal, newVal);    //Update Node
        myList();
        break;
    case 10:
        sortList();                    //Sort List
        myList();
        break;
    case 0:
        dynamicMenu();             //return to Main Menu
        break;
    default:
        cout << "Incorrect Operation\n";
        myList();
        break;
    }
}

void insertAtBeginning(int value)
{
    Node *newNode = new Node();
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

void insertAtEnd(int value)
{
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL)
    {
        head = newNode;
    }
    else
    {
        Node* temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void insertBeforeNode(int beforeValue, int newValue)
{
    Node* newNode = new Node();
    newNode->data = newValue;

    if (head == NULL)
```

```cpp
    {
        cout << "\nThe List Is Empty\n\n";
        system("pause");
        delete newNode;
        return;
    }

    if (head->data == beforeValue)
    {
        newNode->next = head;
        head = newNode;
        return;
    }

    Node* temp = head;
    while (temp->next != NULL && temp->next->data != beforeValue)
    {
        temp = temp->next;
    }

    if (temp->next == NULL)
    {
        cout << "\nValue Not Found In The List\n\n";
        system("pause");
        delete newNode;
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
}

void insertAfterNode(int afterValue, int newValue)
{
    Node* newNode = new Node();
    newNode->data = newValue;

    Node* temp = head;
    while (temp != NULL && temp->data != afterValue)
    {
        temp = temp->next;
    }

    if (temp == NULL)
    {
        cout << "\nValue Not Found In The List\n\n";
        system("pause");
        delete newNode;
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
}

void deleteFirstNode()
{
    if (head == NULL)
    {
        cout << "\nThe List Is Empty\n\n";
        system("pause");
        return;
    }

    Node* temp = head;
    head = head->next;
    delete temp;
}

void deleteLastNode()
{
    if (head == NULL)
    {
        cout << "\nThe List Is Empty\n\n";
        system("pause");
        return;
    }

    if (head->next == NULL)
    {
        delete head;
        head = NULL;
        return;
    }

    Node* temp = head;
    while (temp->next->next != NULL)
    {
        temp = temp->next;
    }

    delete temp->next;
    temp->next = NULL;
}

void deleteValue(int value)
{
    Node* temp = head;
    Node* prev = NULL;

    if (temp != NULL && temp->data == value)
    {
        head = temp->next;
        delete temp;
        return;
    }

    while (temp != NULL && temp->data != value)
    {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) return;

    prev->next = temp->next;
    delete temp;
}

void updateNode(int oldVal, int newVal)
```

```
{
    Node* temp = head;
    while (temp != NULL)
    {
        if (temp->data == oldVal)
        {
            temp->data = newVal;
            return;
        }
        temp = temp->next;
    }
    cout << "\nValue Not Found In The List\n\n";
    system("pause");
}

void sortList()
{
    if (head == NULL || head->next == NULL)
    {
        return;
    }

    Node* i = head;
    Node* j = NULL;
    int temp;

    while (i != NULL)
    {
        j = i->next;
        while (j != NULL) {
            if (i->data > j->data)
            {
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }
            j = j->next;
        }
        i = i->next;
    }
}

void displayList()
{
    Node *temp = head;
    while (temp != NULL)
    {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n\n";
}
///////////////////////////List End/////////////////////////////////


//////////////////////(static Structures)/////////////////////////////////
int* static_arr;
int static_top = -1;
int sizes;
int h = -1;
```

```cpp
int t = -1;
int c = 0;
//***************************
void my_Static_Stack();
void Static_Stack_pop();
void Static_Stack_push(int val);
int Static_Stack_isempty();
int Static_Stack_isfull();
void stack_Stack_print();
//***************************
void my_Static_Liner_Queue();
void Static_Liner_Queue_Push(int val);
void Static_Liner_Queue_Pop();
int Static_Liner_Queue_isempty();
int Static_Liner_Queue_isfull();
void queue_print();
//***************************
void my_Static_Circular_Queue();
void Static_Circular_Queue_Push(int val);
void Static_Circular_Queue_Pop();
int Static_Circular_Queue_isempty();
int Static_Circular_Queue_isfull();
void Circular_queue_print();
//***************************
void staticMenu()
{
    system("cls");
    int select;
        cout << "\n";
    cout << "             Enter The Type Of Structures \n\n";
    cout << "             ----------------------------\n";
    cout << "            |     (static Structures)    |\n";
    cout << "             ----------------------------\n";
    cout << "            | 1: Stack                   |\n";
    cout << "            | 2: Liner Queue             |\n";
    cout << "            | 3: Circular Queue          |\n";
    cout << "             ----------------------------\n\n";
    cout << "Press 0 to go back\n\n";
    cout << "Enter The Transaction Number -> ";
    cin>>select;
    switch(select)
    {
        case 1:
            static_insert_type_val = 1;
            static_insert_type();
            break;
        case 2:
            static_insert_type_val = 2;
            static_insert_type();
            break;
        case 3:
            static_insert_type_val = 3;
            static_insert_type();
            break;
        case 0:
            main();
            break;
        default:
            staticMenu();
            break;
    }
}

void static_insert_type()
{



    system("cls");
    int select;
        cout << "\n";
    cout << "             ----------------------------\n";
    cout << "            |          (Choose)          |\n";
    cout << "             ----------------------------\n";
    cout << "            | 1: Manual Enter            |\n";
    cout << "            | 2: Random                  |\n";
    cout << "             ----------------------------\n\n";
    cout << "Press 0 to go back\n\n";
    cout << "Enter The Number -> ";
    cin>>select;
    switch(select)
    {
        case 1:
            if(static_insert_type_val == 1)
            {
                system("cls");
                cout << "Enter the size of the stack-> ";
                cin >> sizes;
                static_arr = new int[sizes];
                static_top = -1;
                my_Static_Stack();
            }



            else if(static_insert_type_val == 2)
            {
                system("cls");
                cout << "Enter the size of the Linear queue-> ";
                cin >> sizes;
                static_arr = new int[sizes];
                h = -1;
                t = -1;
                my_Static_Liner_Queue();
            }


            else if(static_insert_type_val == 3)
            {
                system("cls");
                cout << "Enter the size of the Circular queue-> ";
                cin >> sizes;
                static_arr = new int[sizes];
                h = -1;
                t = -1;
                c = 0;
                my_Static_Circular_Queue();
            }


```

```cpp
        case 2:
            if(static_insert_type_val == 1)
            {
                system("cls");
                cout << "Enter the size of the stack-> ";
                cin >> sizes;
                static_arr = new int[sizes];
                static_top = -1;
                for (int i = 0; i < sizes; i++)
                {
                    Static_Stack_push(rand()%10+1);
                }
            my_Static_Stack();
            }

            if(static_insert_type_val == 2)
            {
                system("cls");
                cout << "Enter the size of the Linear queue-> ";
                cin >> sizes;
                static_arr = new int[sizes];
                h = -1;
                t = -1;
                for (int i = 0; i < sizes; i++)
                {
                    Static_Liner_Queue_Push(rand()%10+1);
                }
            my_Static_Liner_Queue();
            }

            if(static_insert_type_val == 3)
            {
                system("cls");
                cout << "Enter the size of the Circular queue-> ";
                cin >> sizes;
                static_arr = new int[sizes];
                h = -1;
                t = -1;
                c = 0;
                for (int i = 0; i < sizes; i++)
                {
                    Static_Circular_Queue_Push(rand()%10+1);
                }
            my_Static_Circular_Queue();
            }

            break;
        case 0:
            system("cls");
            staticMenu();
            break;
        default:
            static_insert_type();
            break;
    }
}
//*************************************************

void my_Static_Stack()
{
    system("cls");
    int op , val;
    cout << "   ---Stack size Is "<<sizes<<"---\n\n";
    stack_Stack_print();
    cout << "*********************\n";
    cout << "The Operation Of Stack\n\n";
    cout << "  1->Push\n";
    cout << "  2->Pop\n\n";
    cout << "Press 0 to go back\n\n";
    cout<<"Insert The Operation Number-> ";
    cin >> op;

    switch(op) {
        case 1:
            if(Static_Stack_isfull()==0)
             {
                cout<<"\nInsert The Value : ";
                cin>> val;
                Static_Stack_push(val);
                cout<<"\n";
             }
            else
            {
                cout << "\n";
                cout<<"Stack Is Full\n";
                cout << "\n\n";
                system("pause");
            }

                my_Static_Stack();
                break;

        case 2:
             if(Static_Stack_isempty()==0)
                Static_Stack_pop();
            else
            {
                cout << "\n";
                cout<<"Stack Is Empty\n";
                cout << "\n\n";
                system("pause");

            }
            my_Static_Stack();
            break;

        case 0:
            if(Static_Stack_isempty()==0)
             {
                for (int i = 0; i <= static_top; i++)
                {
                    static_arr[static_top--];
                }
             }

            staticMenu();
            break;
        default:
            cout << "\n";
            cout << "Incorrect Operation\n\n";
```

```cpp
                cout << "\n\n";
                system("pause");
                my_Static_Stack();
                break;
        }
}

int Static_Stack_isfull()
{
    if (static_top == sizes - 1)
        return 1;
    else
        return 0;
}

int Static_Stack_isempty()
{
    if (static_top == -1)
        return 1;
    else
        return 0;
}

void Static_Stack_push(int val)
{
    static_arr[++static_top] = val;
}

void Static_Stack_pop()
{
    cout << "\n";
    cout << "Delete: " << static_arr[static_top--] << "\n";
    cout << "\n";
    system("pause");
    cout << "\n\n";
}

void stack_Stack_print()
{

    if (Static_Stack_isempty())
    {
        cout << "       Stack Is Empty\n\n";
    }
    else
    {
        for (int i = static_top; i >= 0; i--)
        {
            cout <<"          "<< static_arr[i] << "\n";
        }
        cout << "\n";
    }
}
//************************************************

void my_Static_Liner_Queue()
{
    system("cls");
    int op, val;
    cout << " ---Liner Queue size Is "<<sizes<<"---\n\n";
    queue_print();
    cout << "****************************\n";
    cout << "The Operation Of Liner Queue\n\n";
    cout << "  1->Push\n";
    cout << "  2->Pop\n\n";
    cout << "Press 0 to go back\n\n";
    cout << "Insert The Operation Number-> ";
    cin >> op;

    switch (op)
    {
        case 1:
            if (Static_Liner_Queue_isfull() == 0)
            {
                cout << "\nInsert The Value : ";
                cin >> val;
                Static_Liner_Queue_Push(val);
                cout << "\n";
            }
            else
            {
                cout << "\n";
                cout << "Queue Is Full\n";
                cout << "\n\n";
                system("pause");
            }

            my_Static_Liner_Queue();
            break;

        case 2:
            if (Static_Liner_Queue_isempty() == 0)
                Static_Liner_Queue_Pop();
            else
            {
                cout << "\n";
                cout << "Queue Is Empty\n";
                cout << "\n\n";
                system("pause");
            }
            my_Static_Liner_Queue();
            break;

        case 0:
            for(int i= 0; i<=c; i++)
                static_arr[h++];
            h = -1;
            t = -1;
            c = 0;
            staticMenu();
            break;
        default:
            cout << "\n";
            cout << "Incorrect Operation\n\n";
            cout << "\n\n";
            system("pause");
            my_Static_Liner_Queue();
            break;
    }
}
```

```cpp

void Static_Liner_Queue_Push(int val)
{
    if(h==-1)
        h=0;
    static_arr[++t] = val;
    c++;
}

void Static_Liner_Queue_Pop()
{
    if (Static_Liner_Queue_isempty() == 1)
    {
        cout << "\nQueue Is Empty\n";
        system("pause");
    }
    else
    {
        cout << "\nDelete: " << static_arr[h++] << "\n";
        if (h > t) // Reset the queue if all elements are popped
        {
            h = t = -1;
        }
        system("pause");
    }
    c--;
}

int Static_Liner_Queue_isfull()
{
    if (t == sizes - 1)
        return 1;
    else
        return 0;
}

int Static_Liner_Queue_isempty()
{
    return (h == -1 || h > t) ? 1 : 0;
}

void queue_print()
{

    if (Static_Liner_Queue_isempty())
    {
        cout << "      Queue Is Empty\n\n";
    }
    else
    {
        for (int i = h; i <= t; i++)
        {
            cout << "  " << static_arr[i];
        }
        cout << "\n\n";
    }
}
//***************************************************

void my_Static_Circular_Queue()
```

```cpp
{
    system("cls");
    int op, val;
    cout << " ---Circular Queue size Is "<<sizes<<"---\n\n";
    Circular_queue_print();
    cout << "*******************************\n";
    cout << "The Operation Of Circular Queue\n\n";
    cout << "  1->Push\n";
    cout << "  2->Pop\n\n";
    cout << "Press 0 to go back\n\n";
    cout << "Insert The Operation Number-> ";
    cin >> op;

    switch (op)
    {
        case 1:
            if (Static_Circular_Queue_isfull() == 0)
            {
                cout << "\nInsert The Value : ";
                cin >> val;
                Static_Circular_Queue_Push(val);
                cout << "\n";
            }
            else
            {
                cout << "\n";
                cout << "Queue Is Full\n";
                cout << "\n\n";
                system("pause");
            }

            my_Static_Circular_Queue();
            break;

        case 2:
            if (Static_Circular_Queue_isempty() == 0)
                Static_Circular_Queue_Pop();
            else
            {
                cout << "\n";
                cout << "Queue Is Empty\n";
                cout << "\n\n";
                system("pause");
            }
            my_Static_Circular_Queue();
            break;

        case 0:
              h = -1;
            t = -1;
            c = 0;

            staticMenu();
            break;
        default:
            cout << "\n";
            cout << "Incorrect Operation\n\n";
            cout << "\n\n";
            system("pause");
            my_Static_Circular_Queue();
            break;
    }
}

void Static_Circular_Queue_Push(int val)
{
    if (Static_Circular_Queue_isfull())
    {
        cout << "Queue is full, cannot push value.\n";
        return;
    }

    if (h == -1)
        h = 0;

    t = (t + 1) % sizes;
    static_arr[t] = val;
    c++;
}

void Static_Circular_Queue_Pop()
{
    if (Static_Circular_Queue_isempty())
    {
        cout << "Queue is empty, cannot pop value.\n";
        return;
    }

    cout << "Delete: " << static_arr[h] << "\n";
    h = (h + 1) % sizes;
    c--;

    if (c == 0)
    {
        h = -1;
        t = -1;
    }

    system("pause");
}

int Static_Circular_Queue_isempty()
{
    return c == 0;
}

int Static_Circular_Queue_isfull()
{
    return c == sizes;
}

void Circular_queue_print()
{
    if (Static_Circular_Queue_isempty())
    {
        cout << "Queue Is Empty\n\n";
    }
    else
    {
        int i = h;
        for (int count = 0; count < c; count++)
        {
            cout << "  " << static_arr[i];
            i = (i + 1) % sizes;
        }
        cout << "\n\n";
    }
}

//***********************************************

int main()
{
    system("cls");
    int select;
        cout << "\n";
    cout << "             ----------------------------\n";
    cout << "            |          (Choose)          |\n";
    cout << "             ----------------------------\n";
    cout << "            | 1: Dynamic     2: Static   |\n";
    cout << "             ----------------------------\n\n";
    cout << "Press 0 To Stop the program\n\n";
    cout << "Enter The Number -> ";
    cin>>select;
    switch(select)
    {
        case 1:
            dynamicMenu();
            break;
        case 2:
            staticMenu();
            break;
        case 0:
            system("cls");
            return 0;
            break;
        default:
            main();
            break;
    }


}
```