**DATA STRUCTURE**

**Name:** Faizan Tariq

**Roll No:** 22F-3858

**Section:** 3-D

**Assignment # 1**

**TASK 1**

#include <iostream>

using namespace std;

class ArrayADT

{

private:

int\* arr;

int size;

public:

ArrayADT(int size)

{

arr = new int[size]; // creates an array of size dynamically

this->size = size;

for (int i = 0; i < size; i++)

{

cout << "\nEnter Element in index " << i + 1<<" : ";

cin >> arr[i];

}

}

int get(int index) // this function Get the element at a specific index and return to user.

{

if (index <= size && index > 0)

{

return arr[index - 1]; // this will return value in that array

}

else

{

cout << "\nYour given index doesn't exists in array." << endl;

return -1;

}

}

void insert(int index)

{

if (index <= size + 1 && index > 0)

{

int tempElement;

cout << "\nEnter Element To add in array :";

cin >> tempElement;

cout << "\n";

int\* newArray = new int[size + 1]; // now create an array of size increased by one

for (int i = 0; i < index - 1; i++) // copy elements before insertation index

{

\*(newArray+i) = \*(arr+i);

}

newArray[index - 1] = tempElement;

for (int i = index; i <= size; i++)

{

\*(newArray + i) = \*(arr + i - 1); // copies elements of old array in new array after specified index

}

delete[] arr; // deallocate old array elements

size++;

arr = newArray;

}

else

{

cout << "\nYour given index doesn't exists in array." << endl;

}

}

void Delete(int index)

{

if (index <= size && index > 0)

{

int\* newArray = new int[size - 1];

for (int oldIndex = 0, newIndex = 0; oldIndex < size; oldIndex++, newIndex++)

{

if (oldIndex == index - 1)

{

newIndex--;

}

else

{

newArray[newIndex] = arr[oldIndex];

}

}

delete[] arr;

arr = newArray;

size--;

}

else {

cout << "\nIndex out of range, please enter valid index.";

}

}

int Size() // gets length of array

{

return size;

}

void Append()

{

cout << "\nEnter Element to append in array";

int tempElement = 0;

cin >> tempElement;

int\* newArray = new int[size + 1];

for (int i = 0; i < size; i++)

{

newArray[i] = arr[i]; // fills new array with old array

}

newArray[size] = tempElement; // fill element at end of new array

delete[] arr; // de allocates old array

size++;

arr = newArray; // assign address of new array to old array

}

void Display() {

for (int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

}

};

int main()

{

int length;

cout << "Enter Length of array to create :";

cin >> length;

ArrayADT array(length);

int index = 0;

int choice = 0;

while (choice !=7) {

cout << "\n1. Insert Element at specific index.\n";

cout << "2. Get Element from specific index.\n";

cout << "3. Delete Element of specific index.\n";

cout << "4. Append Element in array.\n";

cout << "5. Get size of array.\n";

cout << "6. Display array.\n";

cout << "7. Exit\n";

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter Index :";

cin >> index;

cout << "\n";

array.insert(index);

break;

case 2:

cout << "Enter Index :";

cin >> index;

cout << "\n";

cout << "Your Element at index" << index << " is : " << array.get(index) << endl;

break;

case 3:

cout << "Enter Index :";

cin >> index;

cout << "\n";

array.Delete(index);

break;

case 4:

array.Append();

break;

case 5:

cout << "Size of array : " << array.Size() << endl;

break;

case 6:

array.Display();

break;

default:

break;

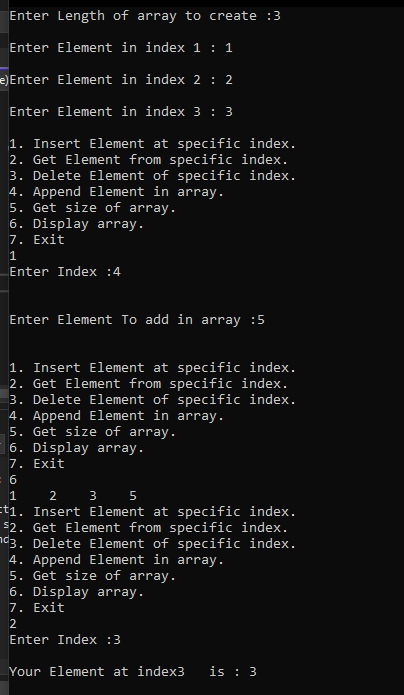
}

}

return 0;

}

OUTPUT :



A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

TASK 1 (PART B)

CODE:

#include <iostream>

using namespace std;

class ArrayADT

{

private:

int\* arr;

int size;

int evenIndex =0, oddIndex =1;

public:

ArrayADT(int size)

{

arr = new int[size]; // creates an array of size dynamically

this->size = size;

for (int i = 0; i < size; i++)

{

cout << "\nEnter Element in index " << i + 1 << " : ";

cin >> arr[i];

}

}

int get(int index) // this function Get the element at a specific index and return to user.

{

if (index <= size && index > 0)

{

return arr[index - 1]; // this will return value in that array

}

else

{

cout << "\nYour given index doesn't exists in array." << endl;

return -1;

}

}

void insert(int index)

{

if (index <= size + 1 && index > 0)

{

int tempElement;

cout << "\nEnter Element To add in array :";

cin >> tempElement;

cout << "\n";

int\* newArray = new int[size + 1]; // now create an array of size increased by one

for (int i = 0; i < index - 1; i++) // copy elements before insertation index

{

\*(newArray + i) = \*(arr + i);

}

newArray[index - 1] = tempElement;

for (int i = index; i <= size; i++)

{

\*(newArray + i) = \*(arr + i - 1); // copies elements of old array in new array after specified index

}

delete[] arr; // deallocate old array elements

size++;

arr = newArray;

}

else

{

cout << "\nYour given index doesn't exists in array." << endl;

}

}

void Delete(int index)

{

if (index <= size && index >= 0)

{

int\* newArray = new int[size - 1];

for (int oldIndex = 0, newIndex = 0; oldIndex < index; oldIndex++, newIndex++)

{

if (oldIndex == index - 1)

{

newIndex--;

}

else

{

newArray[newIndex] = arr[oldIndex];

}

}

delete[] arr;

arr = newArray;

size--;

}

else {

cout << "\nIndex out of range, please enter valid index.";

}

}

int Size() // gets length of array

{

return size;

}

void Append()

{

cout << "\nEnter Element to append in array";

int tempElement = 0;

cin >> tempElement;

int\* newArray = new int[size + 1];

for (int i = 0; i < size; i++)

{

newArray[i] = arr[i]; // fills new array with old array

}

newArray[size] = tempElement; // fill element at end of new array

delete[] arr; // de allocates old array

size++;

arr = newArray; // assign address of new array to old array

}

void Display() {

for (int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

}

void EvenOddInput(int element) {

cout << "\n\"REMEMBER INDEX STARTS FROM 0\"\n";

if (element % 2 == 0) {

if (evenIndex >= size) {

cout << "\nEven Indexes are full.\n";

return;

}

arr[evenIndex] = element;

evenIndex += 2;

}

else {

if (oddIndex >= size) {

cout << "\nOdd Indexes are full.\n";

return;

}

arr[oddIndex] = element;

oddIndex += 2;

}

}

};

int main()

{

int length;

cout << "Enter Length of array to create :";

cin >> length;

ArrayADT array(length);

int index = 0;

bool menu = true;

while (menu) {

cout << "\n1. Insert Element at specific index.\n";

cout << "2. Get Element from specific index.\n";

cout << "3. Delete Element of specific index.\n";

cout << "4. Append Element in array.\n";

cout << "5. Get size of array.\n";

cout << "6. Display array.\n";

cout << "7. EvenODD input.\n";

cout << "8. Exit\n";

int choice = 0;

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter Index :";

cin >> index;

cout << "\n";

array.insert(index);

break;

case 2:

cout << "Enter Index :";

cin >> index;

cout << "\n";

cout << "Your Element at index" << index << " is : " << array.get(index) << endl;

break;

case 3:

cout << "Enter Index :";

cin >> index;

cout << "\n";

array.Delete(index);

break;

case 4:

array.Append();

break;

case 5:

cout << "Size of array : " << array.Size() << endl;

break;

case 6:

array.Display();

break;

case 7:

int tempNumber;

cout << "Enter Even/Odd number to enter in even/odd index :";

cin >> tempNumber;

array.EvenOddInput(tempNumber);

break;

default:

cout << "Exiting\n";

menu = false;

break;

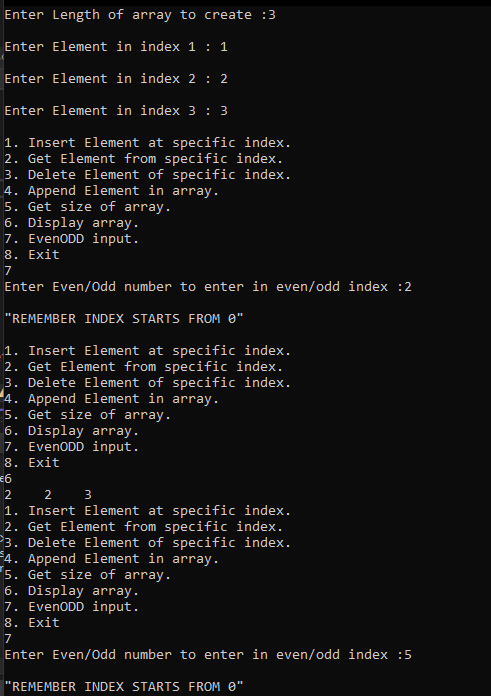
}

}

return 0;

}

OUTPUT:



A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

TASK 1 PART C

CODE :

#include <iostream>

using namespace std;

class ArrayADT

{

private:

int\* arr;

int size;

int evenIndex = 0, oddIndex = 1;

public:

ArrayADT(int size)

{

arr = new int[size]; // creates an array of size dynamically

this->size = size;

for (int i = 0; i < size; i++)

{

cout << "\nEnter Element in index " << i + 1 << " : ";

cin >> arr[i];

}

}

int get(int index) // this function Get the element at a specific index and return to user.

{

if (index <= size && index > 0)

{

return arr[index - 1]; // this will return value in that array

}

else

{

cout << "\nYour given index doesn't exists in array." << endl;

return -1;

}

}

void insert(int index)

{

if (index <= size + 1 && index > 0)

{

int tempElement;

cout << "\nEnter Element To add in array :";

cin >> tempElement;

cout << "\n";

int\* newArray = new int[size + 1]; // now create an array of size increased by one

for (int i = 0; i < index - 1; i++) // copy elements before insertation index

{

\*(newArray + i) = \*(arr + i);

}

newArray[index - 1] = tempElement;

for (int i = index; i <= size; i++)

{

\*(newArray + i) = \*(arr + i - 1); // copies elements of old array in new array after specified index

}

delete[] arr; // deallocate old array elements

size++;

arr = newArray;

}

else

{

cout << "\nYour given index doesn't exists in array." << endl;

}

}

void Delete(int index)

{

if (index <= size && index >= 0)

{

int\* newArray = new int[size - 1];

for (int oldIndex = 0, newIndex = 0; oldIndex < index; oldIndex++, newIndex++)

{

if (oldIndex == index - 1)

{

newIndex--;

}

else

{

newArray[newIndex] = arr[oldIndex];

}

}

delete[] arr;

arr = newArray;

size--;

}

else {

cout << "\nIndex out of range, please enter valid index.";

}

}

int Size() // gets length of array

{

return size;

}

void Append()

{

cout << "\nEnter Element to append in array";

int tempElement = 0;

cin >> tempElement;

int\* newArray = new int[size + 1];

for (int i = 0; i < size; i++)

{

newArray[i] = arr[i]; // fills new array with old array

}

newArray[size] = tempElement; // fill element at end of new array

delete[] arr; // de allocates old array

size++;

arr = newArray; // assign address of new array to old array

}

void Display() {

for (int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

}

void EvenOddInput(int element) {

cout << "\n\"REMEMBER INDEX STARTS FROM 0\"\n";

if (element % 2 == 0) {

if (evenIndex >= size) {

cout << "\nEven Indexes are full.\n";

return;

}

arr[evenIndex] = element;

evenIndex += 2;

}

else {

if (oddIndex >= size) {

cout << "\nOdd Indexes are full.\n";

return;

}

arr[oddIndex] = element;

oddIndex += 2;

}

}

void printDivisors() {

for (int i = 0; i < size; i++)

{

cout << arr[i] << " : ";

for (int j = 1; j <= arr[i] / 2; j++) {

if (arr[i] % j == 0) {

cout << j<<" ";

}

}

cout << endl;

}

}

};

int main()

{

int length;

cout << "Enter Length of array to create :";

cin >> length;

ArrayADT array(length);

int index = 0;

bool menu = true;

while (menu) {

cout << "\n1. Insert Element at specific index.\n";

cout << "2. Get Element from specific index.\n";

cout << "3. Delete Element of specific index.\n";

cout << "4. Append Element in array.\n";

cout << "5. Get size of array.\n";

cout << "6. Display array.\n";

cout << "7. EvenODD input.\n";

cout << "8. Print Divisors.\n";

cout << "9. Exit\n";

int choice = 0;

cin >> choice;

switch (choice)

{

case 1:

cout << "Enter Index :";

cin >> index;

cout << "\n";

array.insert(index);

break;

case 2:

cout << "Enter Index :";

cin >> index;

cout << "\n";

cout << "Your Element at index" << index << " is : " << array.get(index) << endl;

break;

case 3:

cout << "Enter Index :";

cin >> index;

cout << "\n";

array.Delete(index);

break;

case 4:

array.Append();

break;

case 5:

cout << "Size of array : " << array.Size() << endl;

break;

case 6:

array.Display();

break;

case 7:

int tempNumber;

cout << "Enter Even/Odd number to enter in even/odd index :";

cin >> tempNumber;

array.EvenOddInput(tempNumber);

break;

case 8:

array.printDivisors();

break;

default:

cout << "Exiting\n";

menu = false;

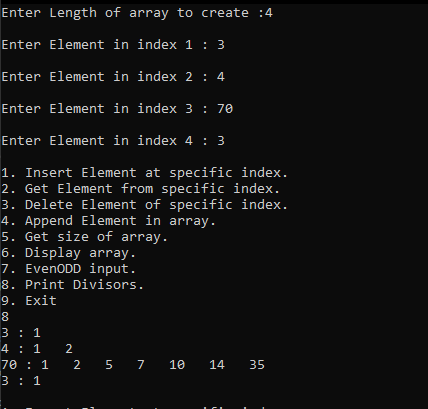
break;

}

}

return 0;

}

OUTPUT :  


TASK 2

CODE :

#include<iostream>

using namespace std;

int main() {

const int size = 10;

int arr[size];

int\* ptr = arr;

cout << "\nEnter elements in your array :" << endl;

for (int i = 0; i < size; ++i) {

cin >> \*(ptr + i);

}

cout << "\nOriginal array elements :" << endl;

for (int i = 0; i < size; ++i) {

cout << \*(ptr + i) << " ";

}

cout << "\nElements having at least two elements greater that that:" << endl;

for (int i = 0; i < size; ++i) {

int count = 0;

for (int j = 0; j < size; ++j) {

if (\*(ptr + j) > \*(ptr + i)) {

count++;

}

}

if (count >= 2) {

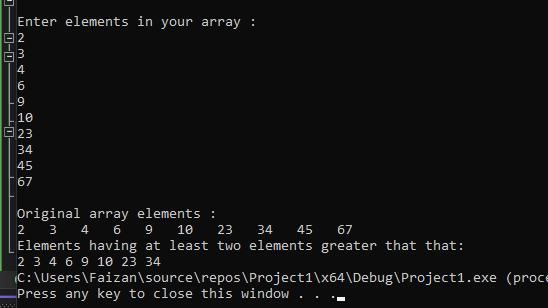
cout << \*(ptr + i) << " ";

}

}

return 0;

}

OUTPUT:  


TASK 3

#include<iostream>

using namespace std;

int main() {

int size = 6;

int\* arr = new int[size];

cout << "Enter ELements in array :\n";

for (int i = 0; i < size; i++)

{

cin >> \*(arr+i);

}

cout << endl;

// display orignal array

cout << "Orignal Array :\n";

for (int i = 0; i < size; i++)

{

cout<< \*(arr+i)<<" ";

}

cout << endl;

int middle = size / 2;

// ascending order sorting before middle

for (int i = 0; i < middle; i++)

{

for (int j = i+1; j < middle; j++)

{

if (\*(arr + i) > \*(arr + j)) {

int temp = \*(arr + i);

\*(arr+i) = \*(arr + j);

\*(arr+j) = temp;

}

}

}

// descending order sorting after middle

for (int i = middle; i < size; i++)

{

for (int j = i + 1; j < size; j++)

{

if (\*(arr + i) < \*(arr + j)) {

int temp = \*(arr + i);

\*(arr + i) = \*(arr + j);

\*(arr + j) = temp;

}

}

}

cout << "Sorted array :\n";

for (int i = 0; i < size; i++)

{

cout << \*(arr+i)<<" ";

}

// now update array using formula x\*(i+1) \* (i-1)

int minIndex = 0, maxIndex = size-1;

int\* tempArr = new int[size];

// copy elements of old array in temp array

for (int i = 0; i < size; i++)

{

\*(tempArr + i) = \*(arr + i);

}

for (int i = 0; i < size; i++)

{

if (i == minIndex) {

\*(tempArr + i) = \*(arr + i) \* \*(arr + maxIndex) \* \*(arr+i+1);

}

else if (i == maxIndex) {

\*(tempArr + i) = \*(arr + i) \* \*(arr + minIndex) \* \*(arr + i - 1);

}

else {

\*(tempArr + i) = \*(arr + i) \* \*(arr + i-1) \* \*(arr + i + 1);

}

}

delete[] arr; // deallocate old memory address of arr

arr = tempArr; // allocate memory address of tempArr to arr

/// print the updated array

cout << "\nUpdated Array :" << endl;

for (int i = 0; i < size; i++)

{

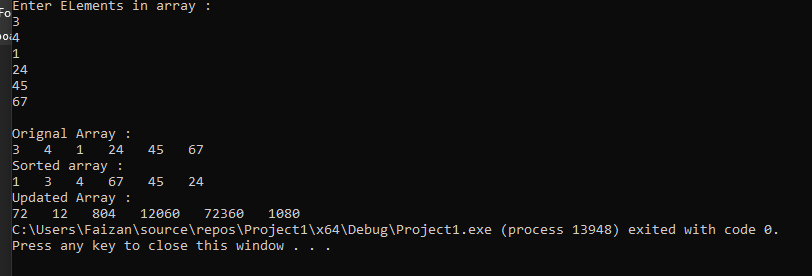
cout << \*(arr + i)<<" ";

}

delete[] arr; // this will deallocate tempArr memory address as well as arr

return 0;

}

**OUTPUT**

**TASK 4**

#include<iostream>

using namespace std;

struct Student {

string name = "";

int id = 0;

float gpa = 0.00;

};

class Student\_Record {

private :

int size;

Student\* records;

public:

Student\_Record() {

records = NULL;

size = 0;

}

void Initialize\_Records() {

cout << "\nEnter Number of Student records:";

cin >> size;

records = new Student[size];

for (int i = 0; i < size; i++)

{

cout << "\nEnter " << i+1 << " Student Record : \n";

cout << "\n Enter Name of Student: ";

cin >> (\*(records+i)).name;

cout << "\n Enter id of Student: ";

cin >> (\*(records+i)).id;

cout << "\n Enter Gpa of Student: ";

cin >> (\*(records + i)).gpa;

}

cout << "\n Records initialized for " << size << " students" << endl;

}

void add\_Record() {

size++;

Student tempStudent;

cout << "\nEnter Student's Name : ";

cin >> tempStudent.name;

cout << "\nEnter Student's id : ";

cin >> tempStudent.id;

cout << "\nEnter Student's gpa : ";

cin >> tempStudent.gpa;

Student\* newRecord = new Student[size];

for (int i = 0; i < size-1; i++) // copying old record

{

\*(newRecord + i) = \*(records + i);

}

\*(newRecord + size-1) = tempStudent;

delete[] records; // de allocate old address of record

records = newRecord; // allocate new address of record

}

void update\_GPA() {

cout << "\nEnter id of student to update GPA: ";

int tempid = 0;

cin >> tempid;

bool idFound = false;

for (int i = 0; i < size && !idFound; i++)

{

if ((\*(records + i)).id == tempid) {

idFound = true;

cout << "\n Enter updated gpa : ";

cin >> (\*(records + i)).gpa;

}

}

if (idFound) {

cout << "\n GPA of student was updated.\n";

}

else {

cout << "\n Cannot find Student with this id.\n";

}

}

void Display\_All\_Records() {

for (int i = 0; i < size; i++)

{

cout << "\nName : " << (\*(records+i)).name;

cout << "\tId : " << (\*(records+i)).id;

cout << "\tGpa : " << (\*(records + i)).gpa << endl;

}

}

void Display\_Highest\_GPA() {

int maxGpa = 0;

for (int i = 0; i < size; i++)

{

if ((\*(records)).gpa > maxGpa) {

maxGpa = (\*(records)).gpa;

}

}

cout << "\nMax Gpa :"<<maxGpa;

}

};

void displayMenu() {

cout << "\n1. Initialize Records\n";

cout << "2. Add Record.\n";

cout << "3. Update Gpa.\n";

cout << "4. Display All Records.\n";

cout << "5. Display Highest Gpa.\n";

cout << "6. Exit";

}

int main() {

cout << "------ STUDENT RECORD MANAGMENT SYSTEM ------\n";

int choice = 0;

int size = 0;

Student\_Record Students;

while (choice!=6) {

displayMenu();

cout << "\n\n Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

Students.Initialize\_Records();

break;

case 2:

Students.add\_Record();

break;

case 3:

Students.update\_GPA();

break;

case 4:

Students.Display\_All\_Records();

break;

case 5:

Students.Display\_Highest\_GPA();

break;

case 6:

break;

default:

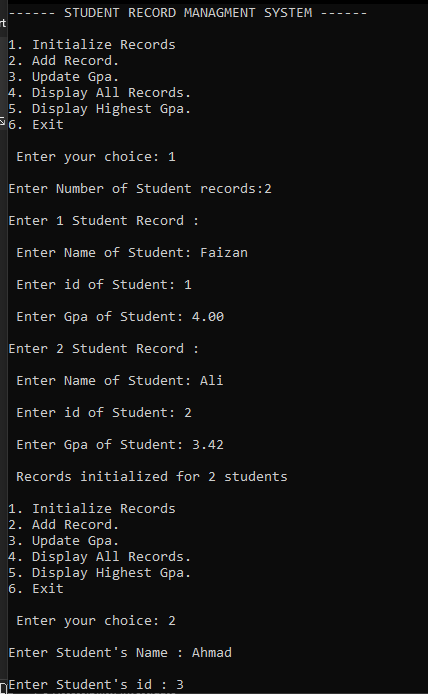
cout << "\nInvalid Choice.\n";

}

}

return 0;

}

**OUTPUT :  
**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**