**Project:**

**Discrete Structure**

**Member Names:** Talha Abdullah Bangyal

Syed Hamza Saeed Kazmi

**Roll Numbers:** 22F-3194

22F-3112

**Section:** BS (AI)-3A

**------------------------------------------------------------------------------**

**Question:**

Take input of 2 sets each of at least three elements. Considering Cartesian. Product of the relation, find whether these relations exist or not. Also find the number of Relations.

1. The Total number of relations

2. Reflexive Relations

3. Symmetric Relations

4. Anti-Symmetric Relations

5. Both Reflexive and Symmetric Relations

6. Both symmetric and Anti-symmetric Relations

**Problem:**

#include<iostream>

using namespace std;

// Part 1: (Cartesian)

int Cartesian(int n1, int n2)

{

int size = 0;

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n2; j++)

{

size++; size++; // this is use to get number of relations

}

}

int n3 = size;

return n3;

}

// Part 2: (Reflexive)

void reflexive(int\* arr1, int\* arr2, int n1, int n2)

{

int Reflexivecheck = 0;

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n2; j++)

{

if (arr1[i] == arr2[j])

{

Reflexivecheck = 1;

break;

}

else {

Reflexivecheck = 0;

}

}

}

// checking reflexive relation

if (Reflexivecheck)

{

cout << "Relation is Reflexive." << endl;

}

else

{

cout << "Relation is not Reflxive." << endl;

}

}

// Part 3: (symmetric)

void symmetric(int\* arr1, int\* arr2, int n1, int n2)

{

int SymmetricCheck = 0;

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n2; j++)

{

if ((arr1[i] == arr2[j] && arr1[i + 1] == arr2[j + 1]) ||

(arr1[i] == arr2[j + 1] && arr1[i + 1] == arr2[j]))

{

SymmetricCheck = 1;

break;

}

else

{

SymmetricCheck = 0;

}

}

}

// checking symmetric relation

if (SymmetricCheck)

{

cout << "Relation is symmetric." << endl;

}

else

{

cout << "Relation is not symmetric." << endl;

}

}

// Part 4: (Anti-symmetric)

void antiSymmetric(int\* arr1, int\* arr2, int n1, int n2)

{

int AntiSymmetricCheck = 1;

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n2; j++)

{

if ((arr1[i] == arr2[j] && arr1[i + 1] == arr2[j + 1]) ||

(arr1[i] == arr2[j + 1] && arr1[i + 1] == arr2[j]))

{

AntiSymmetricCheck = 0;

break;

}

}

}

// checking antisymmetric relation

if (AntiSymmetricCheck)

{

cout << "Relation is anti-symmetric." << endl;

}

else

{

cout << "Relation is not anti-symmetric." << endl;

}

}

// Part 5: (Displaying a Cartesian product)

void display(int\* arr1, int\* arr2, int n1, int n2) {

// use to display a relation

cout << "{";

for (int i = 0; i < n1; i++)

{

for (int j = 0; j < n2; j++)

{

cout << "(" << arr1[i] << "," << arr2[j] << ") ,";

}

}

cout << "}" << endl;

}

// Main runner code

void start()

{

// declareing and initializing a variables

char count;

int size1 = 0, size2 = 0, size3 = 0;

int\* arr1 = nullptr;

int\* arr2 = nullptr;

do

{

// Menu through which user can select and get a specific task

cout << "------------------Menu--------------------" << endl;

cout << "Press 1 to make a Cartesian product" << endl;

cout << "Press 2 to check the total number of relations" << endl;

cout << "Press 3 to check a Reflexive relation" << endl;

cout << "Press 4 to check a Symmetric relation" << endl;

cout << "Press 5 to check an Anti-Symmetric relation" << endl;

cout << "Press 6 to check both Reflexive and Symmetric relations" << endl;

cout << "Press 7 to check both Symmetric and Anti-Symmetric relations" << endl;

cout << "Press 0 to exit" << endl;

cout << "Enter a value: ";

cin >> count;

// use of switch case to go in a specific part

switch (count)

{

case '1':

//Entering the size of sets/array

cout << "Enter size of set 1: ";

cin >> size1;

cout << "Enter size of set 2: ";

cin >> size2;

// using this for to check that user enter a size of array above 0

if (size1 <= 0 || size2 <= 0)

{

cout << "Size of matrix must be greater than 0." << endl;

break;

}

//Creating DMA (Dynamic memory allocation)

arr1 = new int[size1];

arr2 = new int[size2];

//Enter the values of sets/array

cout << "Enter values of set 1: ";

for (int i = 0; i < size1; i++)

{

cin >> arr1[i];

}

cout << "Enter values of set 2: ";

for (int j = 0; j < size2; j++)

{

cin >> arr2[j];

}

// function call

display(arr1, arr2, size1, size2);

size3 = Cartesian(size1, size2);

break;

case '2':

system("CLS");

if (size1 == 0 || size2 == 0)

{

cout << "Relations not created. Please create relation first." << endl;

break;

}

cout << "Total number of relations are: " << size3 / 2 << endl;

cout << "Cartesion product is: " << endl;

// function call

display(arr1, arr2, size1, size2);

break;

case '3':

system("CLS");

if (size1 == 0 || size2 == 0)

{

cout << "Relations not created. Please create relation first." << endl;

break;

}

cout << "Cartesion product is: " << endl;

// function call

display(arr1, arr2, size1, size2);

reflexive(arr1, arr2, size1, size1);

break;

case '4':

system("CLS");

if (size1 == 0 || size2 == 0)

{

cout << "Relations not created. Please create relation first." << endl;

break;

}

cout << "Cartesion product is: " << endl;

// function call

display(arr1, arr2, size1, size2);

symmetric(arr1, arr2, size1, size2);

break;

case '5':

system("CLS");

if (size1 == 0 || size2 == 0)

{

cout << "Relations not created. Please create relation first." << endl;

break;

}

cout << "Cartesion product is: " << endl;

// function call

display(arr1, arr2, size1, size2);

antiSymmetric(arr1, arr2, size1, size2);

break;

case '6':

system("CLS");

if (size1 == 0 || size2 == 0)

{

cout << "Relations not created. Please create relation first." << endl;

break;

}

cout << "Cartesion product is: " << endl;

// function call

display(arr1, arr2, size1, size2);

reflexive(arr1, arr2, size1, size1);

symmetric(arr1, arr2, size1, size2);

break;

case '7':

system("CLS");

if (size1 == 0 || size2 == 0)

{

cout << "Relations not created. Please create relation first." << endl;

break;

}

cout << "Cartesion product is: " << endl;

// function call

display(arr1, arr2, size1, size2);

symmetric(arr1, arr2, size1, size2);

antiSymmetric(arr1, arr2, size1, size2);

break;

case '0':

break;

default:

cout << "Are you Mad, you entered a wrong value!!" << endl;

break;

}

} while (count != '0');

delete[] arr1;

delete[] arr2;

}

//main fucntion

int main() {

start();

return 0;

}

**Outputs:**

**Caption:** In this code **characters** and **floating** numbers can generate an error.

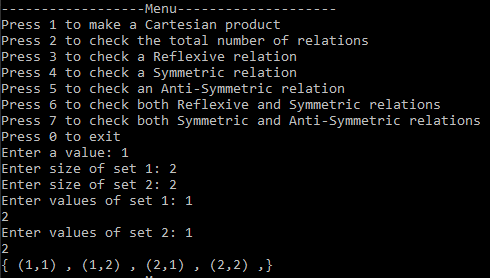
Step 1: Open-Source code

Step 2: Run the code.

Step 3: Output will be shown on console.

Step 4: Press 1 to make a cartesian matrix.

Step 5: Enter size and values in each matrix.



Step 6: Press 2 to get the number of Relations.

A screen shot of a computer

Description automatically generated

Step 7: Press 3 to get the Reflexive Relation.

A screenshot of a computer program

Description automatically generated

Step 8: Press 4 to get the Symmetric Relation.

A screenshot of a computer program

Description automatically generated

Step 9: Press 5 to get the Anti-Symmetric Relation.

A screen shot of a computer

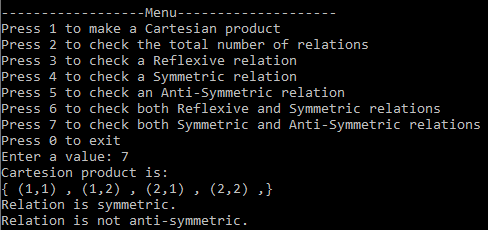
Description automatically generated

Step 10: Press 6 to get the Reflexive and Symmetric Relation.

A screen shot of a computer

Description automatically generated

Step 11: Press 7 to get the Anti-Symmetric and Symmetric Relation.



Step 12: Press 0 to close the program.

A screen shot of a computer

Description automatically generated