***Министерство образования Республики Беларусь***

***Учреждение образования***

***«Брестский государственный технический университет»***

***Кафедра ИИТ***

**Лабораторная работа №3**

**По дисциплине Дискретная математика за II семестр**

**Тема: «Функции»**

**Выполнил:**

Студент группы ИИ-15 (1)

1-го курса

Волк И. А.

**Проверил:**

Глущенко Т.А.

Брест 2018

Цель: познакомиться с функциями.

***Задание 1.***

1. На множествах *B = {1,2,3,4}* и *C = {5,6,7,8}* задано отношение *S*  согласно вашему варианту. Определить (программная реализация) является ли заданное отношение функцией и если да, определить тип функции (инъекция, сюръекция, биекция). Результат проиллюстрировать графически. Варианты заданий указаны в таблице 1.

|  |  |
| --- | --- |
| 7 | S = (1,8),(2,6),(3,8),(4,6) |

*Таблица 1.*

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

using namespace std;

const int notName = system("color f0");

enum typeOfFunc {notFunction = 0, neither, injection, surjection, bijection};

typeOfFunc typeOfFunction(int[][2],int,int\*,int,int\*,int);

bool isFunc(int[][2], int, int\*, int, int\*, int);

bool isInjection(int[][2], int, int\*, int, int\*, int);

bool isSurjection(int[][2], int, int\*, int, int\*, int);

int main()

{

int B[] = {1, 2, 3, 4};

int C[] = {5, 6, 7, 8};

int S[][2] = { {1,8},{2,6},{3,8},{4,6}};

int sizeOfB = sizeof(B)/sizeof(\*B);

int sizeOfC = sizeof(C)/sizeof(\*C);

int sizeOfS = sizeof(S)/sizeof(\*S);

typeOfFunc type = typeOfFunction(S, sizeOfS, B, sizeOfB, C, sizeOfC);

switch (type)

{

case notFunction:

cout << "Not a function." << endl;

break;

case neither:

cout << "Neither." << endl;

break;

case injection:

cout << "Injection." << endl;

break;

case surjection:

cout << "Surjection." << endl;

break;

case bijection:

cout << "Bijection." << endl;

default:

break;

}

system("pause");

return 0;

}

typeOfFunc typeOfFunction(int setOfRel[][2], int sizeOfSetOfRel, int\* A, int sizeOfA, int\* B, int sizeOfB)

{

if (!isFunc(setOfRel, sizeOfSetOfRel, A, sizeOfA, B, sizeOfB))

return notFunction;

else if (isInjection(setOfRel, sizeOfSetOfRel, A, sizeOfA, B, sizeOfB) &&

isSurjection(setOfRel, sizeOfSetOfRel, A, sizeOfA, B, sizeOfB))

return bijection;

else if (isInjection(setOfRel, sizeOfSetOfRel, A, sizeOfA, B, sizeOfB))

return injection;

else if (isSurjection(setOfRel, sizeOfSetOfRel, A, sizeOfA, B, sizeOfB))

return surjection;

else

return neither;

}

bool isFunc(int setOfRel[][2], int sizeOfSetOfRel, int\* A, int sizeOfA, int\* B, int sizeOfB)

{

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

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

if (i != j)

if (setOfRel[i][0] == setOfRel[j][0])

return false;

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

{

bool isInSetOfRel = false;

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

if (A[i] == setOfRel[j][0])

isInSetOfRel = true;

if (!isInSetOfRel)

return false;

}

return true;

}

bool isInjection(int setOfRel[][2], int sizeOfSetOfRel, int\* A, int sizeOfA, int\* B, int sizeOfB)

{

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

{

int numOfRe = 0;

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

if (B[i] == setOfRel[j][1])

numOfRe++;

if (numOfRe > 1)

return false;

}

return true;

}

bool isSurjection(int setOfRel[][2], int sizeOfSetOfRel, int\* A, int sizeOfA, int\* B, int sizeOfB)

{

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

{

int numOfRe = 0;

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

if (B[i] == setOfRel[j][1])

numOfRe++;

if (numOfRe == 0)

return false;

}

return true;

}

Output:



2. Программно реализовать алгоритм сортировки *слиянием* (рекурсивный вариант).

Код программы:

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

#include <cmath>

using namespace std;

void mergeSort(int\*, int);

int main()

{

int sizeOfA = 100;

int \*A = new int[sizeOfA];

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

A[i] = rand() % 1000;

mergeSort(A, sizeOfA);

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

{

for (int j = 0;j < 10&&i < sizeOfA; j++, i++)

cout << A[i] << " ";

cout << endl;

}

system("pause");

return 0;

}

void swap(int\* a, int\* b)

{

int var = \*a;

\*a = \*b;

\*b = var;

}

void mergeSort(int\* arr, int size)

{

int firstSize = size / 2;

if (firstSize == 1)

{

if (size == 3)

{

mergeSort(arr, 2);

mergeSort(arr+1, 2);

mergeSort(arr, 2);

}

if (arr[0] > arr[1])

swap(&arr[0], &arr[1]);

return;

}

mergeSort(arr, firstSize);

mergeSort(arr + firstSize, firstSize);

int \*arrRes = new int[size];

int i = 0, j = firstSize, z = 0;

while (1)

{

if (i == firstSize && j == firstSize+firstSize)

break;

else if (i == firstSize)

{

arrRes[z] = arr[j];

j++; z++;

continue;

}

else if (j == firstSize+firstSize)

{

arrRes[z] = arr[i];

i++; z++;

continue;

}

else

{

if (arr[i] < arr[j])

{

arrRes[z] = arr[i];

z++; i++;

continue;

}

else

{

arrRes[z] = arr[j];

z++; j++;

continue;

}

}

}

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

arr[i] = arrRes[i];

int secondSize = size - firstSize;

if (firstSize != secondSize)

{

mergeSort(arr + firstSize + 1, firstSize);

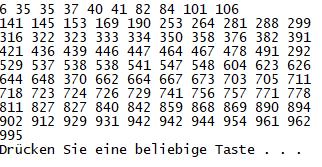
mergeSort(arr + 1, firstSize);

mergeSort(arr, firstSize);

}

}

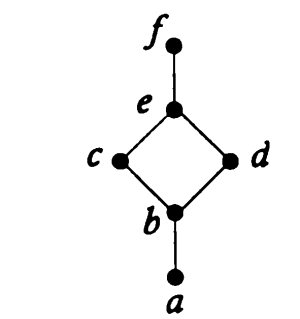
Output:



***Задание 2.***

На подмножестве *A* натуральных чисел *N*,  задано отношение частичного порядка: *a* – делитель *b*. *ЧУ*-множество,  задано диаграммой *Хассе*. Указать:

1. Элементы *ЧУ*- множества .
2. Примеры сравнимых и несравнимых элементов.
3. Наибольший элемент *ЧУ-множества* (если он существует).
4. Наименьший элемент *ЧУ-множества* (если он существует).
5. Максимальный и минимальный элементы *ЧУ-множества*.



1. a – 1, b – 2, c – 8, d – 12, f – 24.

2. Сравнимые: 1, 2; несравнимые: 8, 12.

3. 24 – наибольший

4. 1 – наименьший

5. 24 – максимальный, 1 – минимальный

Вывод: познакомился с функциями.