МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

**ЗВІТ**

з лабораторної роботі №2

з дисципліни “Об'єктивно орієнтоване програмування”

“ДИНАМІЧНІ КЛАСОВІ ТИПИ”

Варіант №15

Виконав:

ст. групи КНТ – 229 Статєв М. В.

Прийняв:

ст. викладач Миронова Н.О.

Запоріжжя

2020

Варіант 15. Створити клас Set для роботи зі структурами типу "Множина". Тип елементу структури обрати самостійно. Множина є набором елементів, кожний з яких має унікальне значення. При додаванні елемента (операція include), який вже є у множині, він не додається. Для цієї структури визначаються операції, звичайні для математичних множин – об’єднання, перетин, віднімання, доповнення. Елементи множини можуть автоматично упорядковуватися, що дає змогу використовувати швидкі операції пошуку елемента з наданим значенням. Передбачити функції-члени класу для виконання таких операцій:

іnclude() – додати новий елемент до множини;

exclude() – вилучити наданий елемент з множини;

union() – об’єднання множин;

inters() – перетин множин;

substr() – віднімання множин;

print() – вивести усі значення множини на екран;

num() – кількість значень множини (потужність).

Код програми:

// main.cpp

#include "set.h"

void Menu();

void Request(Set obj1, Set obj2);

int main() {

Set a;

Set b;

for (;;)

Request(a, b);

}

void Menu() {

cout << "Enter from 0 to 8. What do you want to do?\n";

cout << "1 - Show arrays\n";

cout << "2 - Include element in the array\n";

cout << "3 - Exclude element from array\n";

cout << "4 - Union of two arrays\n";

cout << "5 - Intersection of two arrays\n";

cout << "6 - Substraction arrays\n";

cout << "7 - Symmetric substraction arrays\n";

cout << "8 - Show power of array\n";

cout << "0 - exit\n";

}

void Request(Set obj1, Set obj2){

Menu();

char Answer;

cin >> Answer;

switch (Answer) {

case '1': {

obj1.Output();

obj2.Output();

break;

}

case '2': {

cout << "In which array you want to include?\n";

int Ans;

cin >> Ans;

if (Ans == 1) {

obj1.include();

cout << "It's will array:\n";

obj1.Output();

}

else if (Ans == 2) {

obj2.include();

cout << "It's will array:\n";

obj2.Output();

}

else {

cout << "Enter 1 or 2\n";

}

break;

}

case '3': {

cout << "From which array you want to exclude?\n";

int Ans;

cin >> Ans;

if (Ans == 1) {

obj1.exclude();

cout << "It's will array:\n";

obj1.Output();

}

else if (Ans == 2) {

obj2.exclude();

cout << "It's will array:\n";

obj2.Output();

}

else {

cout << "Enter 1 or 2\n";

}

break;

}

case '4': {

obj1.Union(obj2).Output();

break;

}

case '5': {

obj1.Inters(obj2).Output();

break;

}

case '6': {

obj1.Substr(obj2).Output();

break;

}

case '7': {

obj1.SymSubstr(obj2).Output();

break;

}

case '8': {

obj1.showPower();

obj2.showPower();

break;

}

case '0': {

exit(0);

break;

}

default: {

cout << "Enter from 0 to 8\n";

}

}

}

//set.h

#ifndef SET\_H

#define SET\_H

#include <iostream>

using namespace std;

class Set{

int \*arr = new int [size], size = 0;

public:

Set();

Set(int a);

void Input ();

void Output(); // 1

void include(); // 2

void exclude(); // 3

Set Union(Set obj); // 4

Set Inters(Set obj); // 5

Set Substr(Set obj); // 6

Set SymSubstr(Set obj); // 7

void showPower(); // 8

void Duplicate();

void Sort(int \* arr, int left, int right);

// DEBUG

void ptrOut() {cout << arr;}

void eraseMem() {delete [] arr;}

};

#endif // SET\_H

//set.cpp

#include "set.h"

using namespace std;

void Set::Sort(int \* arr, int left, int right){

int l, r;

int x, y;

l = left; r = right;

x = arr[(left + right) / 2];

do{

while((arr[l] < x) && (l < right)) l++;

while((x < arr[r]) && (r > left)) r--;

if(l<=r){

y = arr[l];

arr[l] = arr[r];

arr[r] = y;

l++;

r--;

}

} while(l<=r);

if(left<r) Sort(arr, left, r);

if(l<right) Sort(arr, l, right);

}

void Set::Duplicate() {

for (int i = 0; i < size - 1; i++){

if (arr[i] == arr[i+1]){

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

arr[j] = arr[j+1];

} size -= 1; i -= 1;

}

}

}

Set::Set() {

size = 0;

Input();

}

Set::Set(int a) {

size += a;

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

arr[i] = ((i\*a) + 1);

}

}

void Set::Input() {

// Enter values in a array

cout << endl << "Enter the array, if you end press '0': ";

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

cin >> arr[i];

if (arr[i] != 0)

size += 1;

else break;

}

Sort(arr, 0, size - 1);

Duplicate();

}

void Set::Output(){

// Show values of array

cout << "\n|<----------------->|\n";

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

cout << arr[i] << ' ';

}

cout << "\n|<--------"<<size<<"--------->|\n";

}

void Set::showPower(){

// Show power of array

cout << "Power of vector = " << size << endl;

}

void Set::include(){

// Add element to array

int NewElement;

cout << "Enter new elemment\n";

cin >> NewElement;

size += 1;

arr[size - 1] = NewElement;

Sort(arr, 0, size - 1);

Duplicate();

}

void Set::exclude() {

// Remove all elements X from array

int ElementToRemove;

cout << "Enter element, you want to remove\n";

cin >> ElementToRemove;

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

if (arr[i] == ElementToRemove) {

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

arr[j] = arr[j+1];

} size -= 1; i -= 1;

}

}

Sort(arr, 0, size - 1);

Duplicate();

}

Set Set::Union(Set obj){

// Union of two arrays

Set res(size + obj.size);

for (int i = 0; i < size; i++) res.arr[i] = arr[i];

for (int i = 0; i < obj.size; i++) res.arr[i + size] = obj.arr[i];

//for (int i = 0; i < res.size; i++) cout << endl<< res.arr[i] << endl;

Sort(res.arr, 0, res.size - 1);

res.Duplicate();

return res;

}

Set Set::Inters(Set obj){

// Intersection of two arrays

Set res(0);

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

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

if (arr[i] == obj.arr[j]) {

res.size += 1;

res.arr[res.size - 1] = arr[i];

}

}

}

Sort(res.arr, 0, res.size - 1);

res.Duplicate();

return res;

}

Set Set::Substr(Set obj){

// Substraction of two arrays

Set res(0);

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

int j = 0;

while(j < obj.size && obj.arr[j] != arr[i])

j++;

if (j == obj.size) {

res.size += 1;

res.arr[res.size - 1] = arr[i];

}

}

Sort(res.arr, 0, res.size - 1);

res.Duplicate();

return res;

}

Set Set::SymSubstr(Set obj){

// Symmetric substraction of two arrays

Set res(0);

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

int j = 0;

while(j < obj.size && obj.arr[j] != arr[i])

j++;

if (j == obj.size) {

res.size += 1;

res.arr[res.size - 1] = arr[i];

}

}

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

int j = 0;

while(j < size && arr[j] != obj.arr[i])

j++;

if (j == size) {

res.size += 1;

res.arr[res.size - 1] = obj.arr[i];

}

}

Sort(res.arr, 0, res.size - 1);

res.Duplicate();

return res;

}