Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Дискретная математика

«Калькулятор множеств»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Рустамханова Г. И.

Пермь 2023

**Постановка задачи:**

Реализовать калькулятор множеств с возможностью:

- задания универсального множества (универсума)

- выбора создания трёх множеств по условию, перечислением и вводом мощности со случайным заполнением.

- выполнения основных операций над множествами (объединение, пересечение, дополнение, разность и симметрическая разность).

**Алгоритм работы:**

В качестве структуры, стоящей в основе множества используется вектор (std::vector). В классе MySet находится реализация множества и операций вставки, вывода и получения элементов.

Класс SetCalculator реализует операции над множествами, также создание множеств путём ручного ввода (manualInput), случайным заполнением (randomFilling) и заполнением при помощи условия (conditionFilling). Кроме того, в классе реализовано вычисление вводимого выражения с помощью алгоритма «сортировочной станции» (shunting yard algorithm).

Операция объединения выполнена с помощью двух не вложенных циклов, первый проходит до первого множества, вставляя все попавшиеся значения, второй - до конца второго множества, вставляя только те значения которых нет в результирующем множестве.

Пересечение реализовано с помощью цикла, проходящего по первому множеству. Если во время проверки было найдено совпадение во втором множестве, в конечное вставляется совпадающее значение.

Операция дополнения реализована с помощью одного цикла. Во время выполнения происходит перемещение по универсуму, и, если текущего числа в универсуме нет, оно вставляется в конечное множество.

Операция разности выполнена при помощи цикла, проходящего по первому множеству. При условии, что элемента первого множества нет во втором, вставляется данный элемент. Операция симметрической разности выполнена похожим образом, только с добавлением второго цикла.

Класс Menu реализует меню, позволяющее пользователю выбирать операцию, способ заполнения множества, замены и вычисления выражения, заданного пользователем.

**Результат тестирования:**

При запуске программы, выводится текст, предлагающий ввести минимальное и максимальное значение универсума. Если при вводе окажется так, что правая граница окажется меньше левой, программа произведёт повторную попытку ввода правой границы.

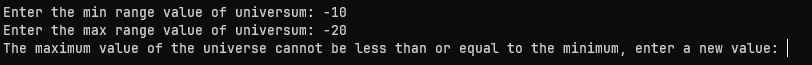


Рис. 1 - Ввод границ универсума

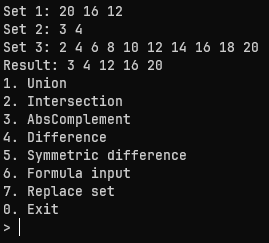
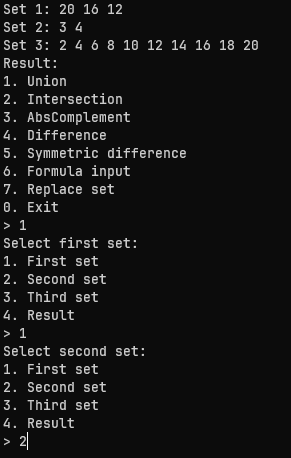


Рис. 2 - Успешное выполнение операции объединения

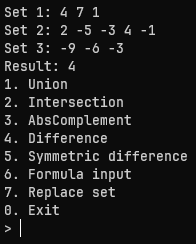


Рис. 2 - Пересечение 1 и 2 множеств

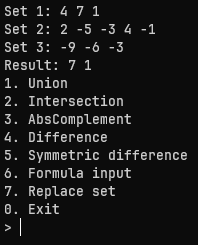


Рис. 2 - Разность 1 и 2 множеств

Елси пользователь введёт несуществующий выбор, программа проигнорирует данный ввод.

Если при формульном вводе пользователь выберет некорректный ответ, программа также предложит ввести новое значение.

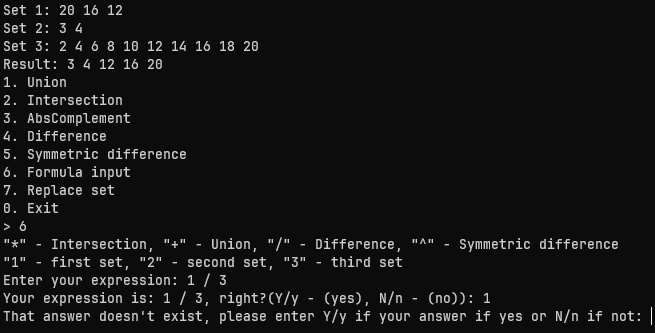


Рис. 3 - Некорректный ввод ответа

Также программа успешно вычисляет значения выражений:

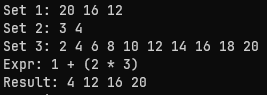


Рис. 4 - Вычисление выражения

**Исходный код:**

Основные операции над множествами:

MySet SetCalculator::Union(MySet& set1, MySet& set2) {

MySet result;

for (int i = 0; i < set1.size(); i++) {

result.AddAtEnd(set1.set[i]);

}

for (int i = 0; i < set2.size(); i++) {

if (!result.Find(set2.set[i])) {

result.AddAtEnd(set2.set[i]);

}

}

sort(result.set.begin(), result.set.end());

return result;

}

MySet SetCalculator::Intersection(MySet& set1, MySet& set2) {

MySet result;

for (int i = 0; i < set1.size(); i++) {

if (set2.Find(set1.set[i])) {

result.AddAtEnd(set1.set[i]);

}

}

return result;

}

MySet SetCalculator::AbsComplement(MySet& set) {

MySet result;

for (int i = minU; i <= maxU; i++) {

if (!set.Find(i)) {

result.AddAtEnd(i);

}

}

return result;

}

MySet SetCalculator::Diff(MySet& set1, MySet& set2) {

MySet result;

for (int i = 0; i < set1.size(); i++) {

if (!set2.Find(set1.set[i])) {

result.AddAtEnd(set1.set[i]);

}

}

return result;

}

MySet SetCalculator::SymDiff(MySet& set1, MySet& set2) {

MySet result;

for (int i = 0; i < set1.size(); i++) {

if (!set2.Find(set1.set[i])) {

result.AddAtEnd(set1.set[i]);

}

}

for (int i = 0; i < set2.size(); i++) {

if (!set1.Find(set2.set[i])) {

result.AddAtEnd(set2.set[i]);

}

}

return result;

}

Функции создания множеств:

void SetCalculator::manualInput(MySet& set) {

int size;

cout << "Enter the size: ";

cin >> size;

while (size <= 0) {

cout << "Size cannot be negative or zero, enter new size: ";

cin >> size;

}

cout << "Enter values\n";

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

int val;

cin >> val;

while (!inUniversum(val)) {

cout << "Value cannot be outside universum, enter new value: ";

cin >> val;

}

set.AddAtEnd(val);

}

}

void SetCalculator::randomFilling(MySet& set) {

int size;

cout << "Enter set power: ";

cin >> size;

while (size <= 0) {

cout << "Size cannot be negative or zero, enter new size: ";

cin >> size;

}

srand(time(NULL));

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

set.AddAtEnd(rand() % (maxU - minU + 1) + minU);

}

}

void SetCalculator::conditionFilling(MySet& set) {

bool isNegative;

int choiceSign;

cout << "Are the elements negative or positive?\n";

cout << "1. Negative\n";

cout << "2. Positive\n> ";

cin >> choiceSign;

while (choiceSign != 1 && choiceSign != 2) {

cout << "That choice doesn't exist, please enter new value: ";

cin >> choiceSign;

}

isNegative = choiceSign == 1;

int multiplicity;

cout << "Enter the multiplicity value: ";

cin >> multiplicity;

while (multiplicity == 0) {

cout << "Invalid input, please enter new value: ";

cin >> multiplicity;

}

int i = minU;

while (i <= maxU) {

if (i % multiplicity == 0) {

if (isNegative) {

if (i < 0) {

set.AddAtEnd(i);

}

}

else {

if (i > 0) {

set.AddAtEnd(i);

}

}

}

i++;

}

}

Основная функция меню:

void Menu::MenuEnter() {

SetCalculator calc;

calc.SetUniversum();

int switchChoice;

MySet setArr[4];

for (int i = 0; i < 3; i++) {

cout << "Set " << i + 1 << ":\n";

calc.FillSet(setArr[i]);

system("cls");

}

MySet newSet;

PrintSetArray(setArr, 3);

string expr;

int firstSet, secondSet;

int setChoice;

do {

Print();

cin >> switchChoice;

cin.ignore();

switch (switchChoice) {

case 1:

firstSet = SelectSet("Select first set:\n");

secondSet = SelectSet("Select second set:\n");

setArr[3] = calc.Union(setArr[firstSet - 1], setArr[secondSet - 1]);

break;

case 2:

firstSet = SelectSet("Select first set:\n");

secondSet = SelectSet("Select second set:\n");

setArr[3] = calc.Intersection(setArr[firstSet - 1], setArr[secondSet - 1]);

break;

case 3:

int choiceSet;

cout << "Which set you want to complement:\n";

cout << "1. First set\n";

cout << "2. Second set\n";

cout << "3. Third set\n> ";

cin >> choiceSet;

while (choiceSet != 1 && choiceSet != 2 && choiceSet != 3) {

cout << "That choice does not exist, please enter new value\n> ";

cin >> choiceSet;

}

switch (choiceSet) {

case 1:

setArr[0] = calc.AbsComplement(setArr[0]);

break;

case 2:

setArr[1] = calc.AbsComplement(setArr[1]);

break;

case 3:

setArr[2] = calc.AbsComplement(setArr[2]);

break;

}

break;

case 4:

firstSet = SelectSet("Select first set:\n");

secondSet = SelectSet("Select second set:\n");

setArr[3] = calc.Diff(setArr[firstSet - 1], setArr[secondSet - 1]);

break;

case 5:

firstSet = SelectSet("Select first set:\n");

secondSet = SelectSet("Select second set:\n");

setArr[3] = calc.SymDiff(setArr[firstSet - 1], setArr[secondSet - 1]);

break;

case 6:

expr = calc.ExpressionInput();

setArr[3] = calc.EvaluateExpr(expr, setArr);

break;

case 7:

setChoice;

cout << "Select set, that you want to replace:\n";

cout << "1. First set\n";

cout << "2. Second set\n";

cout << "3. Third set\n> ";

cin >> setChoice;

while (setChoice != 1 && setChoice != 2 && setChoice != 3) {

cout << "That choice does not exist, please select new set:\n> ";

cin >> setChoice;

}

cout << "Set " << setChoice << ":\n";

calc.FillSet(newSet);

setArr[setChoice - 1] = newSet;

break;

default:

break;

}

system("cls");

PrintSetArray(setArr, 3);

if (expr != "") {

cout << "Expr: " << expr << "\n";

expr = "";

}

cout << "Result: ";

setArr[3].Print();

} while (switchChoice != 0);

}