Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

ОТЧЕТ

к лабораторной работе №2

**«ДРУЖЕСТВЕННЫЕ ФУНКЦИИ И КЛАССЫ. ПЕРЕГРУЗКА ОПЕРАЦИЙ В ЯЗЫКЕ С++»**

|  |  |
| --- | --- |
|  | |
|  |
|  |
|  |
|  |  | | |
|  | | |  | |  |
|  | | |  | |  |
| Студент гр. 324402 | | |  | | Цевелюк А. И. |
| Проверил | | |  | | Булыга Д. И. |

Минск 2024

**1 Цель работы**

Цель работы:понять назначение и необходимость использования дружественных функций и классов, понять необходимость и изучить принципы и особенности перегрузки операторов.

**2 Теоретические сведения по лабораторной работе**

Лабораторная работа посвящена изучению дружественных функций и классов в языке C++. В рамках работы создаются классы с перегруженными операторами, реализовать дружественные функции для доступа к закрытым членам класса, а также изучить особенности перегрузки операторов. Ключевыми аспектами являются создание конструкторов, деструкторов, методов доступа к полям класса, а также реализация friend-функций для операторов ввода/вывода в поток.

**3 Формулировка индивидуального задания**

Создать класс Set, в котором перегрузить операторы для нахождения пересечения и объединения двух множеств. Создать методы доступа к полям класса. Выберите для этих действий подходящие по семантике символы операторов. Память под матрицы выделять в конструкторе, освобождать в деструкторе. Реализовать конструкторы (по умолчанию, с параметрами, копирования), деструктор. Реализовать friend-функции для операторов ввода/вывода в поток для созданного класса.

**4 Код решения индивидуального задания**

#include <algorithm>

#include <iostream>

#include <cstdlib>

#include <string>

class set

{

private:

int\* array\_;

int size\_;

std::string name\_;

public:

set();

set(const int size, const bool random);

set(const int size, const std::string& name);

set(const int size, const std::string& name, const bool random);

set(const set& obj);

~set();

friend void print\_set(const set& set);

friend void sort(const set& set);

friend void write\_to\_file(const set& set);

friend set operator&(const set& left\_operand, const set& right\_operand);

friend set operator|(const set& left\_operand, const set& right\_operand);

};

set::set()

{

this->size\_ = rand() % 15 + 1;

this->array\_ = new int[size\_];

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

this->array\_[i] = rand() % 100;

this->name\_ = "unknown set";

}

set::set(const int size, const bool random)

{

this->size\_ = size;

this->array\_ = new int[size];

this->name\_ = "unknown set";

if (random)

{

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

this->array\_[i] = rand() % 100;

}

else

{

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

{

std::cout << "Enter element of set No" << i + 1 << ": ";

checked\_int\_input(this->array\_[i]);

}

}

}

set::set(const int size, const std::string& name)

{

this->size\_ = size;

this->array\_ = new int[size];

this->name\_ = name;

}

set::set(const int size, const std::string& name, const bool random)

{

this->size\_ = size;

this->array\_ = new int[size];

this->name\_ = name;

if (random)

{

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

this->array\_[i] = rand() % 100;

}

else

{

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

{

std::cout << "Enter element of set \"" + this->name\_ + "\" No" << i + 1 << ": ";

checked\_int\_input(this->array\_[i]);

}

}

}

set::set(const set& obj)

{

size\_ = obj.size\_;

name\_ = obj.name\_;

array\_ = new int[size\_];

for (int i = 0; i < size\_; i++) array\_[i] = obj.array\_[i];

}

set::~set()

{

delete[] array\_;

}

void clear\_input()

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

void checked\_int\_input(int& input)

{

while (!(std::cin >> input))

{

clear\_input();

std::cout << "Enter valid input: ";

}

}

void checked\_unsigned\_input(unsigned& input)

{

int to\_unsigned;

checked\_int\_input(to\_unsigned);

if (to\_unsigned < 0)

to\_unsigned \*= -1;

input = static\_cast<unsigned>(to\_unsigned);

}

void print\_set(const set& set)

{

std::cout << "Set \"" << set.name\_ << "\": ";

for (int i = 0; i < set.size\_; i++)

std::cout << set.array\_[i] << ' ';

std::cout << '\n';

}

set operator&(const set& left\_operand, const set& right\_operand)

{

int return\_size = 0;

const std::string return\_name = left\_operand.name\_ + '&' + right\_operand.name\_;

for (int i = 0; i < left\_operand.size\_; i++)

{

bool temp = false;

for (int j = 0; j < right\_operand.size\_; j++)

{

if (left\_operand.array\_[i] == right\_operand.array\_[j])

{

temp = true;

break;

}

}

if (!temp)

return\_size++;

}

return\_size += right\_operand.size\_;

set return\_set(return\_size, return\_name);

int temp\_it = 0;

for (int i = 0; i < left\_operand.size\_; i++)

{

bool temp = false;

for (int j = 0; j < right\_operand.size\_; j++)

{

if (left\_operand.array\_[i] == right\_operand.array\_[j])

{

temp = true;

break;

}

}

if (!temp)

{

return\_set.array\_[temp\_it] = left\_operand.array\_[i];

temp\_it++;

}

}

for (int i = temp\_it, j = 0; i < return\_size; i++, j++)

return\_set.array\_[i] = right\_operand.array\_[j];

sort(return\_set);

return return\_set;

}

set operator|(const set& left\_operand, const set& right\_operand)

{

int return\_size = 0;

const std::string return\_name = left\_operand.name\_ + '|' + right\_operand.name\_;

for (int i = 0; i < left\_operand.size\_; i++)

{

bool temp = false;

for (int j = 0; j < right\_operand.size\_; j++)

{

if (left\_operand.array\_[i] == right\_operand.array\_[j])

{

temp = true;

break;

}

}

if (temp)

return\_size++;

}

set return\_set(return\_size, return\_name);

int temp\_it = 0;

for (int i = 0; i < left\_operand.size\_; i++)

{

for (int j = 0; j < right\_operand.size\_; j++)

{

if (left\_operand.array\_[i] == right\_operand.array\_[j])

{

return\_set.array\_[temp\_it] = left\_operand.array\_[i];

temp\_it++;

break;

}

}

}

sort(return\_set);

return return\_set;

}

void sort(const set& set)

{

for (int i = 0; i < set.size\_ - 1; i++)

for (int j = 0; j < set.size\_ - i - 1; j++)

if (set.array\_[j] > set.array\_[j + 1])

std::swap(set.array\_[j], set.array\_[j + 1]);

}

void end\_of\_function()

{

system("pause");

system("cls");

}

void write\_to\_file(const set& set)

{

std::fstream file;

file.open("lab2\_logs.txt", std::ios::app);

file << "Set \"" << set.name\_ << "\": ";

for (int i = 0; i < set.size\_; i++)

file << set.array\_[i] << ' ';

file << '\n';

file.close();

}

void clear\_file()

{

std::fstream file;

file.open("lab2\_logs.txt", std::ofstream::out | std::ofstream::trunc);

file.close();

}

int main()

{

srand(static\_cast<unsigned>(time(nullptr)));

clear\_file();

const set set1;

write\_to\_file(set1);

int size\_set2;

std::cout << "Enter size of set No2: ";

checked\_int\_input(size\_set2);

const set set2(size\_set2, true);

write\_to\_file(set2);

int size\_set3;

std::cout << "Enter size of set No3: ";

checked\_int\_input(size\_set3);

clear\_input();

std::string name\_set3;

std::cout << "Enter name of set No3: ";

std::getline(std::cin, name\_set3, '\n');

const set set3(size\_set3, name\_set3, false);

write\_to\_file(set3);

int choice = 0;

do

{

choice = 0;

print\_set(set1);

print\_set(set2);

print\_set(set3);

std::cout << "Choose your option:\n1 - make an intersections of sets\n2 - make an union of sets\n-1 - exit: ";

while (choice != -1 && !(choice > 0 && choice < 3))

{

checked\_int\_input(choice);

if (choice != -1 && !(choice > 0 && choice < 3))

std::cout << "Enter valid input: ";

}

switch (choice)

{

case -1:

break;

case 1:

print\_set(set1 & set2);

print\_set(set1 & set3);

print\_set(set2 & set3);

write\_to\_file(set1 & set2);

write\_to\_file(set1 & set3);

write\_to\_file(set2 & set3);

end\_of\_function();

break;

case 2:

print\_set(set1 | set2);

print\_set(set1 | set3);

print\_set(set2 | set3);

write\_to\_file(set1 | set2);

write\_to\_file(set1 | set3);

write\_to\_file(set2 | set3);

end\_of\_function();

break;

}

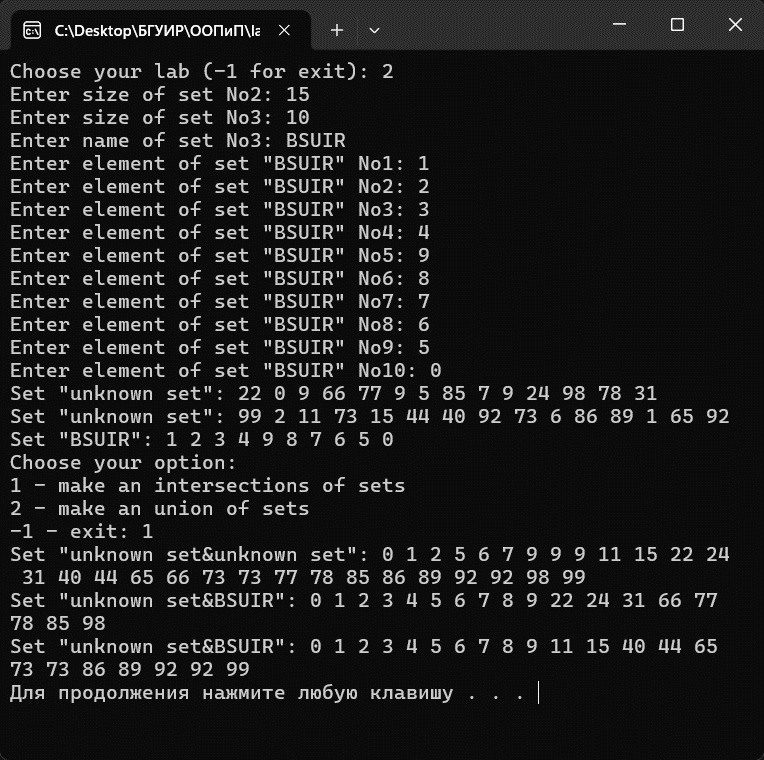
} while (choice != -1);

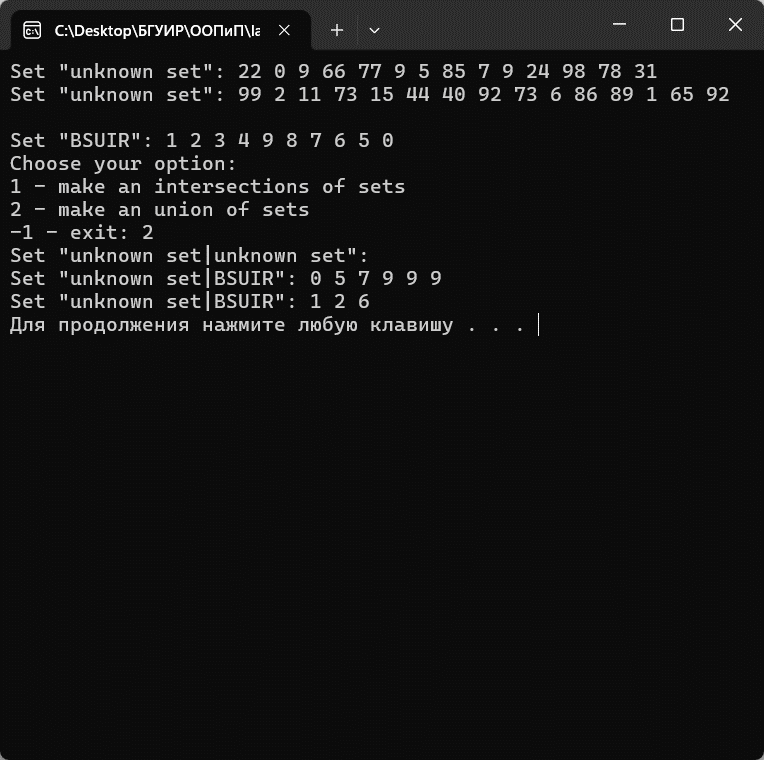
std::cout << "Have a nice day!\n";

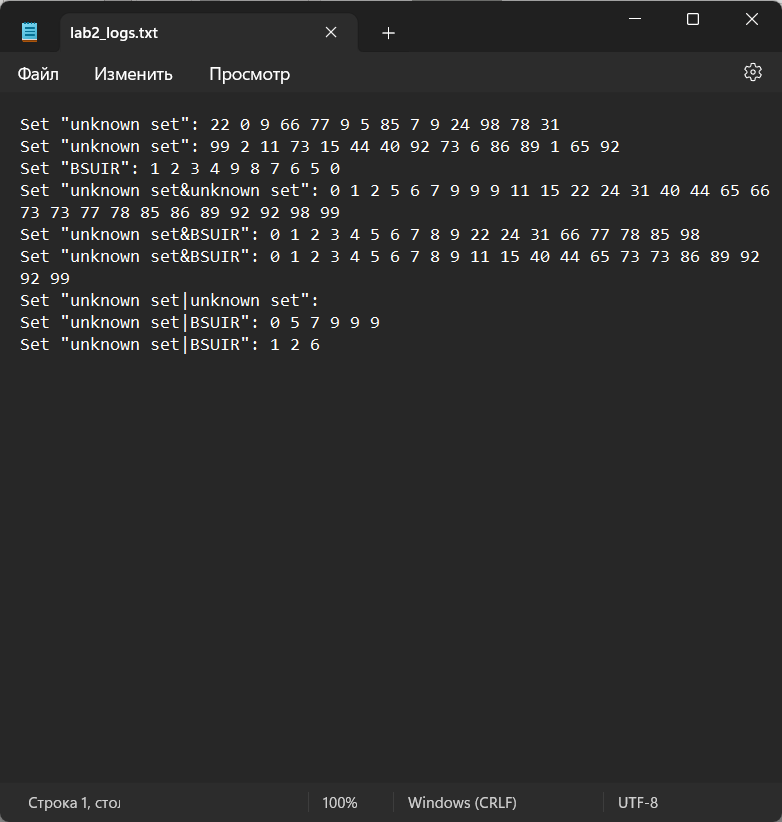
return 0;

}

**5 Скриншоты выполнения индивидуального задания**







**6 Ответы на контрольные вопросы**

1. Почему может потребоваться перегрузка оператора присваивания для класса?

Перегрузка оператора присваивания позволяет определить, как объекты класса будут копироваться или присваиваться друг другу. Это полезно, например, когда класс содержит динамически выделенные ресурсы (например, память), и мы хотим правильно управлять этими ресурсами при присваивании.

1. Можно ли изменить приоритет и арность перегружаемого оператора?

Нет, приоритет и арность перегружаемого оператора не могут быть изменены. Они остаются такими же, как у стандартных операторов.

1. Когда следует переопределять операторы с помощью дружественных функций, а когда с помощью методов класса?

Переопределение операторов с помощью методов класса обычно используется, когда оператор имеет доступ к закрытым членам класса. Дружественные функции используются, когда оператор не имеет доступа к закрытым членам класса, но должен иметь доступ к данным объектов.

1. Назовите особенности использования дружественных функций.

Дружественная функция не является членом класса, но имеет доступ к его закрытым и защищенным членам. Дружественная функция должна быть объявлена внутри класса с ключевым словом friend. Дружественная функция может быть дружественна более чем одному классу. Дружественная функция не наследуется подклассами.

1. Опишите особенности перегрузки постфиксных и префиксных операторов инкремента и декремента.

Постфиксные операторы должны иметь параметр типа int, который не используется, но служит для различения с префиксными операторами. Постфиксные операторы должны возвращать значение объекта до изменения, а префиксные операторы должны возвращать значение объекта после изменения. Постфиксные операторы обычно менее эффективны, чем префиксные, так как они требуют создания временного объекта.

1. Как определить, какую функцию должен выполнять оператор: старую (определенную в языке) или новую (перегруженную для класса)?

Чтобы определить, какую функцию должен выполнять оператор: старую (определенную в языке) или новую (перегруженную для класса), нужно учитывать типы операндов. Если хотя бы один из операндов является объектом класса, для которого перегружен оператор, то будет вызвана перегруженная функция. В противном случае будет использована стандартная операция.

1. Напишите возможную реализацию перегрузки оператора присваивания.

class Array

{

private:

int\* data;

int size;

public:

Array(int n)

{

size = n;

data = new int[size];

}

~Array()

{

delete[] data;

}

Array& operator= (const Array& other)

{

if (this != &other)

{

delete[] data;

size = other.size;

data = new int[size];

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

data[i] = other.data[i];

}

return \*this;

}

};

1. Допускается ли возможность перегрузки операций для встроенных типов данных?

Не допускается возможность перегрузки операций для встроенных типов данных. Перегрузка операторов возможна только для пользовательских типов, таких как классы и перечисления.

1. Целесообразно ли реализовывать перегрузку операторов в классах?

Целесообразность реализации перегрузки операторов в классах зависит от конкретной задачи и целей программиста. В некоторых случаях перегрузка операторов может упростить код, повысить его читаемость и удобство использования, а также сделать классы более естественными и интуитивными. В других случаях перегрузка операторов может привести к путанице, неоднозначности и ошибкам, если операторы используются не по своему назначению или не соответствуют общепринятым соглашениям.

1. Каковы особенности перегрузки унарных и бинарных операторов?

Унарные операторы имеют один параметр, бинарные операторы имеют два параметра. Унарные операторы могут быть перегружены как методы класса или как дружественные функции, бинарные операторы могут быть перегружены только как дружественные функции. Унарные операторы обычно изменяют состояние объекта, бинарные операторы обычно не изменяют состояние объекта, а возвращают новый объект.

**7 Выводы по лабораторной работе**

Вывод: я изучил назначение и необходимость использования дружественных функций и классов, принципы и особенности перегрузки операторов.