Министерство науки и высшего образования РФ

Федеральное государственное автономное образовательное   
учреждение высшего профессионального образования

«Нижегородский государственный университет им. Н.И. Лобачевского»

Институт информационных технологий математики и механики

**Отчет по учебной практике**

**Лабораторная работа № 1**

**«Множества»**

**Выполнил** Ковальчук Александр 3822Б1ПР4

**Проверил:** Лебедев Илья Генадьевич

**Содержание**

[Введение 3](#__RefHeading__3039_1807997502)

[Постановка задачи 4](#__RefHeading__3041_1807997502)

[Руководство пользователя 5](#__RefHeading__3043_1807997502)

[Руководство программиста 6](#__RefHeading__3045_1807997502)

[Описание структуры программы 7](#__RefHeading__3047_1807997502)

[Описание алгоритмов 8](#__RefHeading__3049_1807997502)

[Заключение 9](#__RefHeading__3051_1807997502)

[Литература 10](#__RefHeading__3053_1807997502)

[Приложение 11](#__RefHeading__3055_1807997502)

# **Введение**

Понятие множества, или совокупности, принадлежит к числу простейших математических понятий; оно не определяется, но может быть пояснено при помощи примеров. Так, можно говорить о множестве всех книг, составляющих данную библиотеку, множестве всех точек данной линии, множестве всех решений данного уравнения. Книги данной библиотеки, точки данной линии, решения данного уравнения являются элементами соответствующего множества. Чтобы определить множество, достаточно указать характеристическое свойство элементов, т. е. такое свойство, которым обладают все элементы этого множества и только они. Может случиться, что данным свойством не обладает вообще ни один предмет; тогда говорят, что это свойство определяет пустое множество. То, что данный предмет X есть элемент множества М, записывают так: X ∈ М (читают: X принадлежит множеству М). Если каждый элемент множества А является в то же время элементом множества В, то множество А называется подмножеством, или частью, множества В. Это записывают так: A ⊂ В или В ⊃ А. Таким образом, подмножеством данного множества В является и само множество В. Пустое множество, по определению, считают подмножеством всякого множества. Одной из определяющих характеристик множества является его мощность. В рамках данной работы рассматриваются множества, содержащие конечное число элементов. В этом случае понятие мощности определяется как количество элементов множества. Для таких множеств в математике принята следующая форма записи: A = {a1, a2, …, an}, где А – множество, аi – элементы множества, n – мощность множества. Множество всех возможных элементов называется Универс и обычно обозначается U.

# **Постановка задачи**

# В рамках лабораторной работы ставится задача создания программных средств, поддерживающих эффективное хранение множеств, удовлетворяющих указанным выше допущениям, и выполнение основных операций над множествами:

# включение элемента в множество;

# включение элемента в множество;

# исключение элемента из множества;

# проверка наличия элемента в множестве;

# сложение множеств;

# пересечение множеств;

# разность множеств;

# разность множеств;

# вычисление мощности множества

# **Руководство пользователя**

Данная программа написана в среде разработки Microsoft Visual Studio 2019 на языке С++ и является консольным приложением. При запуске приложения, будет предложено инициализировать размер очереди, число тактов имитации, интенсивность потока задач и производительность процессора.

# **Руководство программиста**

## **Описание структуры программы**

Программа содержит один основной проект – Laba1.

Проект Laba5 содержит в себе модули tbitfield.h, tset.h Модули содержат в себе объявление классов TBitField, Реализация методов классов содержатся в tbitfield.cpp

## **Описание алгоритмов**

**Алгоритмы класса TBitField**

1. TBitField::TBitField(const TBitField& bf) - конструктор копирования
2. TBitField::TBitField(int len) – Конструктор - инициализатор
3. TBitField::~TBitField() - Деструктор.
4. TELEM TBitField::GetMemMask(const int n) const – Создание битовой маски для n
5. int TBitField::GetLength(void) const – Получение длинны бита
6. void TBitField::SetBit(const int n) – Установка бита
7. void TBitField::ClrBit(const int n) – Очистка бита
8. int TBitField::GetBit(const int n) const – Выводит значение бита
9. TBitField& TBitField::operator=(const TBitField& bf) – оператор присваивания
10. int TBitField::operator==(const TBitField& bf) const – оператор сравнения
11. TBitField TBitField::operator|(const TBitField& bf) – операция или
12. TBitField TBitField::operator&(const TBitField& bf) – операция и
13. TBitField TBitField::operator~(void) – присвоение битовому полю отрицательного значения
14. istream& operator>>(istream& istr, TBitField& bf) – ввод
15. ostream& operator<<(ostream& ostr, const TBitField& bf) - вывод

**Алгоритмы класса TSet**

1. TSet::TSet(int mp) : MaxPower(mp), BitField(mp) – Конструктор-инициализатор
2. TSet::TSet(const TSet& s) : MaxPower(s.MaxPower), BitField(s.BitField) – конструктор копирования
3. TSet::TSet(const TBitField& bf) : MaxPower(bf.GetLength()), BitField(bf) – конструктор преобразования типа
4. int TSet::GetMaxPower(void) const – Вывод макс кол-ва элементов
5. int TSet::IsMember(const int elem) const – проверка на принадлежность к элементам множества
6. void TSet::InsElem(const int elem) – включение элемента в множество
7. void TSet::DelElem(const int elem) – исключение элемента из множества
8. TSet& TSet::operator=(const TSet& s) – оператор присваивания
9. int TSet::operator==(const TSet& s) const – оператор сравнения
10. TSet TSet::operator+(const TSet& s) – оператор объединения
11. TSet TSet::operator+(const int elem) – объединение с элементом
12. TSet TSet::operator-(const int elem) – разность с элементом
13. TSet TSet::operator\*(const TSet& s) – оператор пересечения
14. TSet TSet::operator~(void) – оператор дополнения

# **Заключение**

Данная лабораторная работа познакомила нас с битовыми полями и множествами, научила нас применять арифметические операции на множествах, сравнивать множества и пересекать их между собой.

# **Литература**

1. http://old.itmm.unn.ru/most/obuchenie/uchebnye-materialy/

# **Приложение**