Отчет по лабораторной работе №1

Предмет: «Алгоритмы и структуры данных»

Тема лабораторной работы: «Множества на основе битовых полей»

Выполнил:

Студент

Курс: 2

Группа: 3824Б1ФИ2

ФИО: Большаков Владислав Владимирович

2025 г.

Содержание:

1. Постановка задачи
2. Описание класса TBitField, методов класса TBitField
3. Описание класса TSet, методов класса TSet
4. Описание тестов Google Test

1. **Постановка задачи**

   Задача: разработка структуры данных для хранения множеств с использованием битовых полей на языке программирования C++, написание автоматических тестов Google Test для проверки корректности работы реализованных классов. Структурой данных для хранения множеств является класс TSet, представляющий собой реализацию конечного множества. Класс TSet использует механизм агрегации, применяя класс TBitField, который предоставляет возможность хранения характеристического вектора множества.

2. **Описание класса TBitField, методов класса TBitField**

   Класс TBitField предназначен для хранения набора битов (характеристического вектора множества). Каждый бит соответствует определенному элементу множества и может принимать два значения 0 или 1. Единица означает, что элемент принадлежит множеству, ноль – отсутствие элемента в множестве.

   Данный класс содержит три поля: BitLen, pMem, MemLen.

   Описание полей:

   - BitLen – максимальной число битов, которое поддерживает данной объект класса TBitField. Данное поле представлено типом int.
   - pMem – динамический массив типа TELEM (в данной лабораторной работе тип TELEM определен как тип unsigned int)
   - MemLen – размер массива pMem. Представлено типом int.

   Описание методов.

   Private методы:
   GetMemIndex (const int n) const
   - Описание параметров и возвращаемых значений: возвращает значение типа int, принимает константный параметр типа int, является константным методом. Возвращаемое значение – номер элемента, в котором находится бит с индексом n
   - Описание реализуемого функционала: если индекс n не корректен (меньше нуля, больше или равен полю BitLen), то бросается исключение **out_of_range**. С помощью цикла while определяем в какой степени должна быть двойка, чтобы получилось количество бит в элементе TELEM (в нашем случае число бит в TELEM равно 32. Степень двойки равна 5). После делаем операцию n>>5 (эквивалентно n/32).
   - Оценка сложности: степень двойки вычисляется с помощью цикла while. Цикл происходит пока a<32 (начальное значение a=1). За каждую итерацию a увеличивается в 2 раза. Нужно определить за сколько шагов (k) a будет больше или равно 32. Т.е. решить уравнение $2^k \geq 32$. Получаем log(32) (или же log(bitsInElem)). Итоговая оценка: т.к. bitsInElem – фиксированное число, то получаем O(1).

   GetMemMask (const int n) const
   - Описание параметров и возвращаемых значений: возвращает значение типа TELEM (маску для бита n), принимает константный параметр типа int (индекс бита), является константным методом.

- Описание реализуемого функционала: в методе бросается такое же исключение, как и в методе GetMemIndex. Вычисляется индекс бита в элементе. Производится операция 1<<(индекс бита в элементе). По итогу получаем маску, в которой все нули, кроме бита с индексом n.

Public методы:

Конструкторы и деструктор.
TBitField (int len) - конструктор инициализатор.

- Описание параметров и возвращаемых значений: принимает целочисленный параметр типа int.
- Описание реализуемого функционала: создаёт объект с указанным числом бит (len). На основе параметра len происходит расчет значений для поля MemLen. Если параметр len меньше нуля, то бросается исключение **invalid_argument**. Если len равен 0, то полю pMem присваивается nullptr.

TBitField (const TBitField &bf) – конструктор копирования.

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект данного класса.
- Описание реализуемого функционала: копирует состояние другого объекта в объект, для которого вызван данный конструктор (т.е. копируются значения полей: BitLen, MemLen, pMem (копируются значения из ячеек массива, адреса памяти не копируются)). Если BitLen равно 0, то полю pMem присваивается nullptr.
- Оценка сложности: с помощью цикла for осуществляется проход по всему массиву pMem (происходит копирование массива pMem). 1 операция инициализации (присваивания), MemLen операций инкремента, MemLen+1 операций сравнения (i < bf.MemLen), 2*MemLen операций обращения к элементам, MemLen операций присваивания (в теле цикла). Если сложить, то получим 5*MemLen + 2. Итоговая оценка: O(MemLen)

~TBitField () – деструктор.

- Описание реализуемого функционала: освобождает память, выделенную для массива pMem. После освобождения присваивает указателю pMem значение nullptr.

Доступ к битам:

GetLenght ():

- Описание параметров и возвращаемых значений: возвращает тип int. Является константным методом.
- Описание реализуемого функционала: возвращает длину битового поля (т.е. значение BitLen).

SetBit (const int n):

- Описание параметров и возвращаемых значений: возвращает тип void. Принимает константный параметр типа int, являющийся индексом бита.
- Описание реализуемого функционала: устанавливает бит с индексом n в значение 1. Если индекс является некорректным (т.е. принимает отрицательные значения, или значения равные или превышающие BitLen), то бросается исключение **out_of_range**. Также бросается исключение **runtime_error**, если поле pMem принимает значение nullptr. Использует приватные методы: **GetMemIndex**, **GetMemMask**.

ClrBit (const int n):

- Описание параметров и возвращаемых значений: возвращает тип void. Принимает константный параметр типа int, являющийся индексом бита.
- Описание реализуемого функционала: устанавливает бит с индексом n в значение 0. Бросаемые исключение идентичны исключениям, бросаемым в методе SetBit. Использует приватные методы: **GetMemIndex**, **GetMemMask**.

GetBit (const int n) const:

- Описание параметров и возвращаемых значений: возвращает значение типа int. Является константным методом, принимает константный параметр типа int, который является индексом бита.

- Описание реализуемого функционала: возвращает значение бита с индексом n. Бросаемые исключение идентичны исключениям, бросаемым в методе SetBit. Использует приватные методы: **GetMemIndex**, **GetMemMask**.

Теоретико-множественные операции.

Перегруженный **оператор = =**

- Описание параметров и возвращаемых значений: возвращает значение типа int. Принимает константную ссылку на объект данного класса. Является константным методом.

- Описание реализуемого функционала: сначала проверяются значения полей BitLen, затем производится поэлементное сравнение массивов pMem. Если объекты равны возвращается 1. Иначе 0.

- Оценка сложности: поэлементное сравнение осуществляется с помощью цикла for. 1 операция инициализации (присваивания), MemLen операций инкремента, MemLen +1 операций сравнения (i < bf.MemLen), 2*MemLen операций обращения к элементам, MemLen операций сравнения на неравенство. Если сложить, то получим 5*MemLen+2. Итоговая оценка O(MemLen).

Перегруженный **оператор !=**

- Описание параметров и возвращаемых значений: аналогично оператору = =.

- Описание реализуемого функционала: аналогично оператору = =, только если объекты не равны, то возвращается 1. Иначе 0.

- Оценка сложности: аналогично оператору = =. Итоговая оценка O(MemLen).

Перегруженный **оператор присваивания**

- Описание параметров и возвращаемых значений: возвращает ссылку на текущий объект (*this), принимает константную ссылку на объект.
- Описание реализуемого функционала: Копирует состояние объекта в текущий объект (в объект слева от =). Имеется проверка на самоприсваивание (в этом случае возвращается *this). Также если объект имеет нулевую длину битового поля (BitLen = 0), то указателю pMem присваивается nullptr. Возвращение ссылки на текущий объект позволяет делать запись «объект_1 = объект_2 = объект_3 = …» возможной.
- Оценка сложности: вычисление сложности производится также, как и для конструктора копирования. Итоговая оценка: O(MemLen)

Перегруженный **оператор |** (операция «или»).

- Описание параметров и возвращаемых значений: возвращает объект данного класса (новый объект, содержащий результат объединения двух полей). Принимает константную ссылку на объект.
- Описание реализуемого функционала: есть два объекта (текущий объект и объект, с которым будет производится операция «или». Пусть второй называется объект A). Выбирается наибольшая из длин двух объектов. Создается новый объект с этой длиной. Далее текущий объект копируется в новый. После происходит побитовое «или» над ячейками соответствующих массивов (массивы нового объекта и объекта A). Если pMem текущего объекта принимает значение nullptr, то возвращается объект A. Если pMem объекта A принимает значение nullptr, то возвращается текущий объект.
- Оценка сложности: имеется два отдельных цикла, которые проходят по максимальному из двух полей MemLen. Т.к. циклы не являются вложенными, то итоговая оценка: O(*MemLen), где *MemLen (максимальное значение поля MemLen)

Перегруженный **оператор &** (операция «и»).

- Описание параметров и возвращаемых значений: аналогично оператору |, только возвращается новый объект, содержащий результат пересечения двух полей.
- Описание реализуемого функционала: аналогично оператору |. Но выбирается наименьшее из двух полей MemLen, и над ячейками массивов выполняется побитовое &.
- Оценка сложности: O(*MemLen), где *MemLen – наименьшее из полей двух объектов.

Перегруженный **оператор ~**

- Описание параметров и возвращаемых значений: возвращает новый объект, который является результатом побитового отрицания над ячейками исходного массива
- Описание реализуемого функционала: создается новый объект с BitLen как у исходного объекта.

Если поле равно 0, то бросается исключение **runtime_error**. Далее происходит проход по массиву pMem. Выполняется побитовое отрицание к каждому элементу массива. Новые значения элементов присваиваются новому объекту. Далее вычисляем количество битов, которые мы используем в последнем элементе. Если таковые заполняют не весь последний элемент, то применяем специальную маску. Не использованные биты перейдут в нулевое значение.

- Оценка сложности: основная работа – проход по массиву с помощью цикла for. Итоговая оценка O(MemLen).

Перегруженный **оператор ввода**

- Описание параметров и возвращаемых значений: принимает два параметра. Первый – ссылка на объект типа **std::istream**, второй – ссылка на объект данного класса. Возвращает ссылку на объект istream (поэтому возможно применить несколько операторов подряд: **std::cin**>>объект_1>>объект_2).
- Описание реализуемого функционала: из потока ввода считывает строка символов, представляющих собой битовое поле. Если длина данной строки не равна длине битового поля, то бросается исключение **runtime_error**. Далее идет проверка корректности символов строки. Если есть символы отличные от 0 и 1, то бросается исключение **invalid_argument**. С помощью методов **SetBit** и **GlrBit** идет установка битов справа налево (т.к. биты нумеруются справа налево, от младших к старшим).
- Оценка сложности: имеется два цикла for, используемых для прохода по строке. Пусть длина строки равна n. Т.к. сложность методов SetBit и ClrBit равна O(1), данные циклы не являются вложенными, то итоговая сложность равна O(n).

Перегруженный **оператор вывода**

- Описание параметров и возвращаемых значений: принимает два параметра. Первый – ссылка на объект типа **std::ostream**, второй – ссылка константный объект данного класса. Возвращает ссылку на объект ostream (поэтому возможно применить несколько операторов подряд: **std::cout**<<объект_1<<объект_2)
- Описание реализуемого функционала: если длина битового поля равна 0, то бросается исключение **runtime_error**. Цикл for проходит с конца битового поля, каждый бит представляется как символ и записывается в поток (в объект типа ostream).
- Оценка сложности: сложность метода GetBit равна O(1). Биты приводятся к типу char с помощью операции приведения типов static_cast (сложность O(1)). Поэтому, из-за цикла for, итоговая сложность: O(BitLen).

3. **Описание класса TSet, методов класса TSet**

Класс TSet представляет собой реализацию структуры данных «множество». Данный класс использует объект класса TBitField для хранения признаков принадлежности элементов конкретному множеству.

Данный класс содержит два поля: MaxPower, BitField.
Описание полей:

- MaxPower – мощность универсального множества для конкретного множества (в множестве не может быть больше элементов, чем в универсальном множестве).

Данное поле представлено типом int.

- BitField – объект класса TBitField. Данный объект является характеристическим вектором данного множества (в множестве не хранятся элементы. Только признак - есть ли элемент в множестве или нет). Если элемент из универса под индексом n принадлежит множеству, то в битовом поле данный элемент описан как 1. Иначе 0. Нумерация элементов в универсе начинается с 0 ($U = \{u_0, u_1, u_2 \ldots\}$)

Описание методов

Private методы: отсутствуют

Public методы:

Конструкторы и деструктор.
TSet (int mp) – конструктор инициализатор

- Описание параметров и возвращаемых значений: принимает целочисленный параметр типа int
- Описание реализуемого функционала: инициализирует члены класса. MaxPower устанавливается равным значению параметра mp (максимальная мощность универса). Объект класса TBitField создается с длиной равной параметру mp (максимальное количество элементов, которое может хранить множество). Если параметр меньше нуля, то бросается исключение **invalid_argument**.

TSet (const TSet& s) – конструктор копирования

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект данного класса.
- Описание реализуемого функционала: копирует состояние другого объекта в объект, для которого вызван данный конструктор. Т.е. происходит копирование полей MaxPower, BitField.
- Оценка сложности: для поля BitField вызывается конструктор копирования класса TBitField. Из-за этого сложность оценивается как O(MemLen) (сложность конструктора копирования класса TBitField).

Деструктор: для освобождения памяти, отведенной для поля BitField используется деструктор класса TBitField.

TSet (const TBitField& bf) – конструктор преобразования типа из TBitField

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект класса TBitField
- Описание реализуемого функционала: получает длину битового поля (bf.GetLenght()) и инициализирует данной величиной поле MaxPower. Инициализирует поле BitField параметром.
- Оценка сложности: аналогично конструктору **TSet (const TSet& s)** сложность: O(MemLen).

operator TBitField () – оператор преобразования типа к битовому полю.

- Описание параметров и возвращаемых значений: возвращает объект класса TBitField
- Описание реализуемого функционала: метод предназначен для преобразования объекта класса TSet к объекту класса TBitField.
- Оценка сложности: в реализации возвращается объект BitField (return BitField). Из-за этого вызывается конструктор копирования класса TBitField. Значит сложность O(MemLen)

Доступ к битам

GetMaxPower(void) const:

- Описание параметров и возвращаемых значений: возвращает значение типа int. Является константным методом.
- Описание реализуемого функционала: возвращает значение поля MaxPower

InsElem (const int Elem):

- Описание параметров и возвращаемых значений: принимает константный параметр типа int (данный параметр является номером элемента из универса, который нужно включить в наше множество)
- Описание реализуемого функционала: использует метод SetBit() класса TBitField. Бит с индексом Elem будет равен 1. Если поле MaxPower равно 0, то бросается исключение **runtime_error**. Если параметр Elem меньше 0, больше или равен полю MaxPower, то бросается исключение **out_of_range**.

DelElem(const int Elem):

- Описание параметров и возвращаемых значений: принимает константный параметр типа int (данный параметр является номером элемента, который нужно удалить из нашего множества)
- Описание реализуемого функционала: использует метод ClrBit() класса TBitField. Бит с индексом Elem будет равен 0. Точно такие же исключения, как в методе InsElem.

IsMember(const int Elem) const:

- Описание параметров и возвращаемых значений: принимает константный параметр типа int (нужно проверить, есть элемент под номером Elem в множестве или нет). Возвращает параметр типа int. Если есть возвращает 1, иначе 0. Является константным методом.
- Описание реализуемого функционала: использует метод GetBit() класса TBitField. Данный метод возвращает значение бита под индексом Elem. Точно такие же исключения, как в методе InsElem.

Теоретико-множественные операции

Перегруженный **оператор = =**

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект данного класса. Является константным методом. Возвращает значение типа int (1 - если данные множества равны, 0 – в противном случае)
- Описание реализуемого функционала: сравнивает на равенство поля MaxPower, BitField.
- Оценка сложности: для сравнения полей BitField используется перегруженный «оператор = =» класса TBitField. Из-за этого сложность оценивается как O(MemLen)

Перегруженный **оператор ! =**

- Описание параметров и возвращаемых значений: аналогично оператору = = данного класса. Только 1 возвращается, если данные множества не равны. 0 в противном случае.
- Описание реализуемого функционала: производится логическое отрицание результата оператора = =
- Оценка сложности: аналогично оператору = =, т.е. O(MemLen)

Перегруженный **оператор присваивания**:

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект данного класса, возвращает ссылку на текущий объект (*this) (из-за этого возможна запись: объект_1=объект_2=объект_3)
- Описание реализуемого функционала: полям исходного объекта присваиваются значения полей другого объекта. Имеется проверка на самоприсваивание.

- Оценка сложности: используется перегруженный оператор присваивания класса TBitField (для поля BitField). Из-за этого сложность оценивается как O(MemLen)

Перегруженный **оператор + (объединение с элементом):**

- Описание параметров и возвращаемых значений: принимает константный параметр типа int. Возвращает объект данного класса.
- Описание реализуемого функционала: создается новый объект – копия данного объекта (т.е. применяется конструктор копирования данного класса). С помощью метода InsElem добавляется элемент под индексом Elem. Если поле MaxPower исходного множества равно 0, то бросается исключение **runtime_error**. Если индекс Elem меньше 0, больше или равен значению MaxPower, то бросается исключение **out_of_range**.
- Оценка сложности: из-за конструктора копирования сложность оценивается как O(MemLen).

Перегруженный **оператор – (разность с элементом):**

- Описание параметров и возвращаемых значений: аналогично оператору объединения с элементом
- Описание реализуемого функционала: аналогично оператору объединения с элементом, только вместо метода InsElem используется метод DelElem.
- Оценка сложности: аналогично оператору объединения с элементом сложность оценивается как O(MemLen).

Перегруженный **оператор + (объединение двух множеств)**

- Описание параметров и возвращаемых значений: принимает константную ссылку на объект данного класса. Возвращает объект данного класса.
- Описание реализуемого функционала: создается новый объект (поле MaxPower нового объекта равно полю MaxPower текущего объекта). Объединяются битовые поля двух множеств с помощью оператора |. Результат записывается в поле нового объекта BitField. Если поля MaxPower двух множеств не равны, то бросается исключение **runtime_error**. Имеется проверка на самоприсваивание

- Оценка сложности: в методе выполняются операции объединения двух битовых полей (сложность O(MemLen)), присваивание результата в поле BitField нового объекта (вызывается конструктор копирования класса TBitField, сложность O(MemLen)), возвращение в качестве результата нового объекта класса TSet (вызывается конструктор копирования класса TSet, сложность O(MemLen)). Итоговая сложность O(3Memlen), что равно O(MemLen).

Перегруженный **оператор * (пересечение двух множеств)**

- Описание параметров и возвращаемых значений: аналогично оператору объединения двух множеств
- Описание реализуемого функционала: аналогично оператору объединения двух множеств. Но вместо операции | используется оператор пересечения двух битовых полей (т.е. оператор & класса TBitField).
- Оценка сложности: аналогично оператору объединения двух множеств. Итоговая оценка O(MemLen)

Перегруженный **оператор ~ (дополнение к исходному множеству)**

- Описание параметров и возвращаемых значений: возвращает объект данного класса.
- Описание реализуемого функционала: создается копия текущего множества. Для поля BitField применяется оператор ~. Возвращается дополнение исходного множества.
- Оценка сложности: вызывается конструктор копирования данного класса (сложность O(MemLen)), оператор ~ из класса TBitField (сложность O(MemLen)), конструктор копирования класса TBitField (сложность O(MemLen)), конструктор копирования данного класса. Получаем сложность O(4*MemLen), т.е. итоговая сложность O(MemLen).

Перегруженный **оператор ввода**

- Описание параметров и возвращаемых значений: аналогично оператору класса TBitField.
- Описание реализуемого функционала: проверяется значение поля MaxPower. Если значение данного поля равно 0, то бросается исключение **runtime_error**. Данные считываются из потока ввода целиком в переменную типа std::string с помощью функции std::getline(). Далее полученная строка передается в объект класса std::stringstream. С помощью цикла while происходит разделение строки на элементы (разделение происходит по символу “ , “). Каждый элемент проверяется на корректность:
  - ❖ длина элемента не равна 0, т.к. возможно поступление строки вида: «1,2,,,,3,4». В противном случае бросается исключение **runtime_error**.
  - ❖ Элемент должен состоять из символов «0,1,2,3…9» (здесь символы указаны через запятую. В элементе запятых нет). Если присутствуют другие символы, то бросается исключение **invalid_argument**.
  - ❖ Если длина элемента больше нуля и второй символ равен 0, то бросается исключение **invalid_argument.**

- ❖ (перед следующей проверкой элемент приводится к целочисленному типу с помощью функции std::stoi()) Элемент представляет собой индекс элемента универса, который нужно включить в множество. Если данный элемент (индекс) меньше нуля, больше или равен значение поля MaxPower, то бросается исключение **out_of_range**.

Если элемент прошел все проверки, то с помощью метода InsElem он добавляется в множество.

- Оценка сложности: пусть длина вводимой строки равна L, количество элементов, разделенных запятыми, равно K. Считывание с помощью getline, копирование в объект std::stringstream – O(L) (по отдельности). Цикл while повторяется K раз (также вызывается функция getline, которая по итогу проходит по всей строке), K раз используется метод InsElem (также есть цикл for). Итоговая оценка O(L+K).

Перегруженный **оператор вывода**

- Описание параметров и возвращаемых значений: аналогично оператору ввода в классе TBitField
- Описание реализуемого функционала: если поле MaxPower равно 0, ты бросается исключение **runtime_error**. Данный оператор выводит множество в следующем виде: сначала идет битовый вектор (используется перегруженный оператор класса TBitField), после двоеточия в фигурных скобках через запятую представлены индексы элементов универса, которые включены в множество. Используется цикл for, который проходит по каждому элементу множества и с помощью метода IsMember проверяет наличие элемента в множестве.
- Оценка сложности: пусть MaxPower = P. Тогда цикл for повторяется P раз, также используется оператор вывода для класса битовое поле (его сложность также равна O(P), но данный оператор идет перед циклом). Значит итоговая сложность O(P).

4. **Описание тестов Google Test**

Тесты для класса TBitField.

В группе **Test_TBitField0** проверяется корректность работы конструктора инициализации (создание объекта нулевой длины, положительной длины, отрицательной длины) и конструктора копирования класса TBitField (успешность копирования существующего объекта, проверка независимости копий друг от друга, копирование пустого объекта, копирование объекта с большой длиной).

В группе **Test_TBitField1** проверяется корректность работы методов GetLenght(), SetBit(), ClrBit(), GetBit(). Рассмотрены случаи выхода индекса за допустимые границы (обработка ошибок), попытка применения методов к объекту с невыделенной памяти.

В группе **Test_TBitField2** проверяется корректность битовых операций (оператора = =, оператора !=, оператора присваивания, операции «или», операции «и», операции отрицания)

Операции сравнения: рассмотрены случаи совпадения и не совпадения длин полей, совпадения всех битов и отличия хотя бы в одном, равенства объектов с невыделенной памятью.
Оператор присваивания: рассмотрены случаи самоприсваивания, присваивания объекта с невыделенной памятью, применения нескольких операторов подряд (ну и просто корректность работы данного оператора).
Операция «или», операция «и»: работа данных операций над полями разной длины (включая ситуацию, когда длина одного из полей равна 0), одинаковой длины.
Операция отрицания: работа на полях разной длины, включая нулевую длину.

В группе **Test-TBitField3** проверяется корректность работы операторов ввода и вывода.

Оператор ввода: тестируется ввод строк, представляющих собой битовое поле, обработка некорректных символов, применение нескольких операторов подряд, обработка ситуации, связанной с неправильной длиной относительно заданного поля.

Оператор вывода: проверяется корректность отображения состояния битов поля в виде строки, обработка ситуации пустой длины и последовательный вывод нескольких полей.

Тесты для класса TSet.
В группе тестов **Test_Tset0** проверяется корректность работы конструктора инициализации, конструктора копирования, конструктора преобразования типа, операции преобразования типа к битовому полю.

Конструктор инициализации: создание объекта, попытка создания объекта с отрицательной длиной.
Конструктор копирования: проверяется правильной копирования объекта.
Конструктор преобразования типа: проверка корректности работы (копирование мощности и битового поля).
Операция преобразования типа к битовому полю: корректность приведения объекта класса TSet к объекту класс TBitField.

В группе тестов **Test_Tset1** проверяется корректность работы методов GetMaxPower(), InsElem(), DelElem(), IsMember(). Рассмотрены случаи выхода индекса за допустимые границы, применение методов к объекту с невыделенной памятью.

В группе тестов **Test_Tset2** проверяется корректность теоретико-множественных операций.

Операторы сравнения: проверяется корректность работы с применением метода InsElem.

Оператор присваивания: проверяется корректность работы с применением метода InsElem, операторов сравнения. Также проверяется возможность применения данного оператора несколько раз подряд.

Операция объединения множества с элементом, операция разности множества с элементом: корректность работы данных операций, обработка попытки оперировать с индексом элемента вне допустимого диапазона.

В группе тестов **Test_Tset3** проверяется корректность работы операций объединения двух множеств, пересечения двух множеств, дополнения к исходному множеству.

Операции объединения и пересечения двух множеств: проверяется корректность данных операций, обработка попытки применить данные операции к объектам класса TSet разной длины, к объектам с нулевой длиной.

Дополнение к исходному множеству: корректность работы данной операции на объектах класса TSet разной длины.

В группе тестов **Test_Tset4** проверяется корректность работы операторов ввода и вывода.

Оператор ввода: обработка ситуации ввода некорректных символов; ввод символов, которые больше мощности универса или меньше 0.

Оператор вывода: корректность вывода множества, вывод нескольких объектов класса TSet, применение данного оператора к объекту с нулевой длиной.