ФГАОУВО «Южно-Уральский государственный университет (НИУ)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине «Объектно-ориентированное программирование»

Автор работы

студент группы ЕТ-211

_____ А.А.Потава

_____2022 г.

Работа зачтена с оценкой

_____

_____А.К.Демидов

_____2022 г.

Челябинск, 2022

1 Постановка задачи

I. Определить класс-шаблон с использованием динамического распределения памяти согласно варианту и необходимые конструкторы и операции, включая конструктор копий, операция присваивания и если указано операцию индексации. При выходе за границу, переполнении и т.п. вызвать исключительную ситуацию (определить собственные классы) для информирования программы, вызвавшей метод.

11. класс множество элементов заданного типа (для которого заданы операции сравнения на равенство), размером не более указанного в параметрах конструктора. Каждое значение в множестве встречается не более одного раза.
удаление произвольного значения -=, добавление нового значения +=, проверка принадлежности элемента множеству, разность (-) множеств.

При определении друзей класса-шаблона использовать следующий пример
II. Реализовать main с тестами
(создание объектов и выполнение действий с ними, в т.ч. действие, приводящее к возникновению исключительной ситуации, которую необходимо перехватить)

2 Описание интерфейса классов

```
   struct seterror { // базовый класс для ошибок
      virtual ~seterror() {} // деструктор
      virtual const char *what() const=0; // сообщение для
печати
   };

   struct setempty: seterror {
      const char *what() const {return "Множество пустое";}
// сообщение для печати
   };

   struct setfull: seterror {
      const char *what() const {return "Множество
переполнено";} // сообщение для печати
   };

   struct setsmall: seterror {
      const char *what() const {return "Множество слишком
мало для копирования";} // сообщение для печати
   };

   struct setdiferent: seterror {
      const char *what() const {return "Множества разных
размеров";} // сообщение для печати
   };

   struct sethave: seterror {
```

```
      const char *what() const {return "Элемент уже есть в
множестве";} // сообщение для печати
   };

   struct sethavent: seterror {
      const char *what() const {return "Элемента нет в
множестве";} // сообщение для печати
   };

   template <typename T>
   class Set {
      T *a; // указатель на данные в множестве
      int col, // текущее количество
            size; // максимальный размер
   public:
      Set(int size): col(0), size(size),a(new T[size]) {} //
конструктор
      Set(const Set<T> &);// конструктор копий
      ~Set() throw() {delete []a;}// деструктор
      Set<T> &operator=(const Set<T> &); // операция
присваивания
      Set<T> &operator+=(const T &); // операция добавления в
стек
      Set<T> &operator-=(T &); // операция извлечения из
стека
      bool operator==(const Set<T> &); // операция сравнения
множеств
      bool operator==(const T &); // операция сравнения
множеств
      Set<T> operator-(const Set<T> &); // операция разность
множеств
   };
```

3 Описание тестов для проверки классов

```
   int main()
   {
      Set<int> obj(10);
      cout<<"Тест 1. Добавление\n";
      try {
         for (int i=0;; i++) {
            obj+=i;
         }
      }
      catch (seterror &e) {
         cout<<e.what();
      }
```

```
      Set<int> obj1(9);
      cout<<"\nТест 2. Копирование\n";
      try {
         obj1 = obj;
      }
      catch (seterror &e) {
         cout<<e.what();
      }
      Set<int> obj2(11);
      try {
         obj2 = obj;
      }
      catch (seterror &e) {
         cout<<e.what();
      }

      cout<<"\nТест 3. Сравнение и нахождение в множестве
элемента\n";
      try {
         cout << (obj2 == obj) << "\n";
         obj2 += 10;
         cout << (obj2 == obj) << "\n";
         cout << (obj == 9) << "\n";
         cout << (obj == 10) << "\n";
      }
      catch (seterror &e) {
         cout<<e.what();
      }

      cout<<"\nТест 4. Разность и удаление элементов из
множества\n";
      Set<int> obj3(11);
      try {
         obj3 = (obj2-obj);
         for(int i = 10;;i++)
         {
            obj3 -= i;
            cout << i << ' ';
         }
      }
      catch (seterror &e) {
         cout<<e.what();
      }

      return 0;
   }
```

Полученные результаты

Тест 1. Добавление
Множество переполнено
Тест 2. Копирование
Множество слишком мало для копирования
Тест 3. Сравнение и нахождение в множестве элемента
1
0
1
0

Тест 4. Разность и удаление элементов из множества
10 Множество пустое

4 Листинг реализации класса

```
   template <typename T>
   Set<T>:: Set(const Set <T> &c):a(new
T[c.size]),col(c.col),size(c.size) {
      for (int i=0; i<col; i++)
         a[i]=c.a[i];
   }

   template <typename T>
   Set<T> &Set<T>::operator=(const Set<T> &s)
   {
      if(s.col > size) throw setsmall();
      for(int i = 0; i < s.col; i++)
      {
         a[i] = s.a[i];
      }
      col = s.col;
      return *this;
   }

   template <typename T>
   Set<T> &Set<T> ::operator+=(const T &x) {
      if (col==size) throw setfull();
      for(int i = 0; i < col; i++)
      {
         if(a[i] == x) throw sethave();
      }
      a[col++]=x;
      return *this;
   }
```

```
template <typename T>
Set <T> &Set<T>::operator-=(T &x) {
   if (col==0) throw setempty();
   for(int i = 0; i < col; i++)
   {
      if(a[i] == x)
      {
         for(int k = i+1; k < col; k++) a[k-1] = a[k];
         col--;
         return *this;
      }
   }
   throw sethavent();
}

template <typename T>
bool Set<T>::operator==(const Set<T> &s) // операция
сравнения множеств
{
   if(col != s.col) return false;
   for(int i = 0; i < col; i++)
   {
      bool flag = false;
      for(int k = 0; k < s.col; k++)
      {
         if(a[i] == s.a[k]) flag = true;
      }
      if(!flag) return false;
   }
   return true;
}

template <typename T>
bool Set<T>::operator==(const T &x) // операция есть
значение в множестве?
{
   for(int i = 0; i < col; i++)
   {
      if(a[i] == x) return true;
   }
   return false;
}

template <typename T>
Set<T> Set<T>::operator-(const Set<T> &s) // операция
разность множеств
```

```
{
   Set<T> b(size);
   for(int i = 0; i < col; i++)
   {
      bool flag = true;
      for(int k = 0; k < s.col; k++)
      {
         if(a[i] == s.a[k]) flag = false;
      }
      if(flag) b+=a[i];
   }
   return b;
}
```