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

_____ В.А.Малахов

_____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.

6. класс упорядоченный набор элементов заданного типа (для которого заданы операции сравнения)
удаление произвольного элемента по номеру, добавление << нового элемента, получение i-го элемента последовательности с помощью операции индексации.

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

   struct seqempty: seqerror {
      const char *what() const {return "Последовательность
пуста";} // сообщение для печати
   };

   struct seqhavent: seqerror {
      const char *what() const {return "В последовательности
не найдена ячейки под этим индексом";} // сообщение для
печати
   };

   template <typename T>
   class Sequence {
      T *a; // указатель на данные в стеке
```

```
    int col, // текущее количество
         size;
  public:
    Sequence(): col(0), size(1), a(new T[1]) {} //
конструктор
    Sequence(const Sequence<T> &); // конструктор копий
    ~Sequence() throw() {delete []a;} // деструктор
    Sequence<T> &operator<<(const T &value); // операция
добавления в стек
    Sequence<T> &del(int); // удаление по ключу
    T &operator[](int); // поиск по ключу
    T operator[](int) const; // поиск по ключу
  };
```

3 Описание тестов для проверки классов

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

    cout<<"\nТест 2. Получение по индексу\n";
    try {
      for (int i=0;; i++) {
        cout << (obj[i]) << ' ';
      }
    }
    catch (seqerror &e) {
      cout<<"\n"<<e.what();
    }

    cout<<"\nТест 3. Удаление\n";
    try {
      for (int i=0;;) {
        obj.del(i);
      }
    }
    catch (seqerror &e) {
      cout<<"\n"<<e.what();
```

```
    }
    return 0;
}
```

Полученные результаты

Тест 1. Добавление

Тест 2. Получение по индексу
0 1 2 3 4 5 6 7 8 9
В последовательности не найдена ячейки под этим индексом
Тест 3. Удаление

Последовательность пуста

4 Листинг реализации класса

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

template <typename T>
Sequence<T> &Sequence<T>::operator<<(const T &value)
{
    if(col == size)
    {
        T *b = new T[col];
        for(int i = 0; i < col; i++)
        {
            b[i] = a[i];
        }
        delete []a;
        a = new T[size*2];
        size*=2;
        int k = -1;
        for(int i = 0; b[i] < value and i < col; i++)
        {
            k = i;
            a[i] = b[i];
        }
        a[k+1] = value;
        for(int i = k+1; i < col; i++)
        {
            a[i+1] = b[i];
        }
```

```
      col++;
      delete []b;
   }
   else
   {
      int k = col;
      for(int i = col; a[i-1] > value and i > 0; i--)
      {
         k = i-1;
         a[i] = a[i-1];
      }
      a[k] = value;
      col++;
   }
   return *this;
}

template <typename T>
Sequence<T> &Sequence<T>::del(int k)
{
   if(col == 0) throw seqempty();
   if(k < 0 or k >= col) throw seqhavent();
   for(int i = k; i < col-1; i++)
   {
      a[i] = a[i+1];
   }
   col--;
   return *this;
}

template <typename T>
T &Sequence<T>::operator[](int k)
{
   if(k < 0 or k >= col) throw seqhavent();
   return a[k];
}

template <typename T>
T Sequence<T>::operator[](int k) const
{
   if(k < 0 or k >= col) throw seqhavent();
   return a[k];
}
```