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-213

_____ Д.В.Исрафилова

_____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.

13. класс стек элементов заданного типа, размером не более указанного в параметрах конструктора,
добавление << и извлечение >> элемента

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

   struct stackempty: stackerror {
      const char *what() const {return "Стек пуст";} //
сообщение для печати
   };

   struct stackfull: stackerror {
      const char *what() const {return "Стек полон";} //
сообщение для печати
   };

   struct stacksmall: stackerror {
      const char *what() const {return "Стек слишком мал для
копирования";} // сообщение для печати
   };

   template <typename T>
   class Stack {
```

```
    T *a; // указатель на данные в стеке
    int col, // текущее количество
          size; // максимальный размер
public:
    // конструктор
    Stack(int size): col(0), size(size),a(new T[size]) {}
    // конструктор копий
    Stack(const Stack<T> &);
   // деструктор
    ~Stack() throw() {delete []a;}
    // операция присваивания
    Stack<T> &operator=(const Stack<T> &);
    // операция добавления в стек
    Stack<T> &operator<<(const T &);
    // операция извлечения из стека
    Stack<T> &operator>>(T &);
};
```

3 Описание тестов для проверки классов

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

    cout<<"\nТест 2. Присваивание\n";
    Stack<int> objcop(10);
    try {objcop = obj;}
    catch (stackerror &e) {
       cout<<"\n"<<e.what();
    }
    try {
       while (1) {
          int temp;
          objcop>>temp;
          cout<<temp << ' ';
       }
    }
    catch (stackerror &e) {
```

```
        cout<<"\n"<<e.what();
    }

    cout<<"\nТест 3. Извлечение\nИзвлекаем из стека:\n";
    try {
        while (1) {
            int temp;
            obj>>temp;
            cout<<temp << ' ';
        }
    }
    catch (stackerror &e) {
        cout<<"\n"<<e.what();
    }
    return 0;
}
```

Полученные результаты

Тест 1. Добавление
Стек полон
Тест 2. Присваивание
9 8 7 6 5 4 3 2 1 0
Стек пуст
Тест 3. Извлечение
Извлекаем из стека:
9 8 7 6 5 4 3 2 1 0
Стек пуст

4 Листинг реализации класса

```
template <typename T>
Stack<T>:: Stack(const Stack <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>
Stack<T> &Stack<T>::operator=(const Stack<T> &s)
{
    if(s.col > size) throw stacksmall();
    col = s.col;
    for(int i = 0; i < s.col; i++)
    {
        a[i] = s.a[i];
```

```
    }
    return *this;
}

template <typename T>
Stack<T> &Stack<T> ::operator<<(const T &x) {
    if (col==size) throw stackfull();
    a[col++]=x;
    return *this;
}

template <typename T>
Stack <T> &Stack<T>::operator>>(T &x) {
    if (col==0) throw stackempty();
    x=a[--col];
    return *this;
}
```