Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчет по лабораторной работе №5**

**Дисциплина:** Низкоуровневое программирование

**Тема:** программирование на языке Cи

Выполнил

студент гр. 3530901/90003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руднев А.К.

(подпись)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексюк А.О.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Санкт-Петербург   
2021

**Содержание**

[**1. Техническое задание** 3](#_Toc71459406)

[**2. Метод решения** 3](#_Toc71459407)

[**3. Описание библиотеки:** 3](#_Toc71459408)

[**4. Листинг** 4](#_Toc71459409)

[**5. Ошибки и коды ошибок:** 8](#_Toc71459410)

[**6. Программа и методика тестирования:** 8](#_Toc71459411)

[**6.1 Ошибки** 8](#_Toc71459412)

[**6.2 Программа тестирования** 9](#_Toc71459413)

[**6.3 Рисунки выполнения программы без ошибок:** 10](#_Toc71459414)

[**7. Вывод:** 10](#_Toc71459415)

# **1. Техническое задание**

Разработать статическую библиотеку, реализующую set с помощью однонаправленного линейного списка. Разработать демонстрационную программу – консольное приложение для обеспечения ввода, обработки и вывода данных.

# **2. Метод решения**

**Set** — это контейнер, который хранит в себе целочисленные значения в единичном экземпляре.

Реализация будет осуществлена с помощью однонаправленного линейного списка, то есть каждый элемент (узел) хранит информацию, а также ссылку на следующий элемент. Последний элемент списка ссылается на NULL.

1) Первый элемент ссылается на NULL или на второй элемент

2) Последний элемент ссылается на NULL

2) Нет повторяющихся элементов

# **3. Описание библиотеки:**

|  |
| --- |
| Рис. 1 - Описание библиотеки |

1) makeSet() – создание set, где первый элемент ссылается на NULL.

2) add() – добавление элемента в массив.

3) removeElement() – удаление элемента из set.

4) print() – печать set.

5) deleteAll() – удаление всего set.

6) contains() – определение наличия элемента в set.

# **4. Листинг**

|  |
| --- |
| #include "console.h" #include "test.h"   int main() {  console();  //doTest();  return 0; }  **Рис. 2 – main.c** |
| #include <stdio.h> #include <string.h> #include <stddef.h> #include <stdlib.h> #include <ctype.h>  #include "SET.h" #include "console.h" #include "error.h"  #define BUFFER\_SIZE 255  int console() {  work();  return 0; }   void work() {  Set \* set = makeSet();  char buffer[BUFFER\_SIZE];  while (fgets(buffer, BUFFER\_SIZE, stdin)) {  removeSpace(buffer);  if (strcmp(buffer, "#") == 0) break;  char \*command = strtok(buffer, " ");  char \*token;  if (strcmp(command, "insert") == 0) {  while ((token = strtok(NULL, " "))) {  if (!isdigit(\*token)) {  error(2);  } else {  int arg = (int) strtol(token, NULL, 10);  insert(arg, set);  }  }  } else if (strcmp(command, "remove") == 0) {  while ((token = strtok(NULL, " "))) {  if (!isdigit(\*token)) {  error(2);  } else {  int arg = (int) strtol(token, NULL, 10);  removeElement(arg, set);  }  }  } else if (strcmp(command, "contains") == 0) {  while ((token = strtok(NULL, " "))) {  if (!isdigit(\*token)) {  error(2);  } else {  int arg = (int) strtol(token, NULL, 10);  contains(arg, set);  }  }  } else if (strcmp(command, "print") == 0) {  print(set);  } else if (strcmp(command, "deleteAll") == 0) {  deleteAll(set);  } else {  error(1);  }  }  deleteAll(set); }   static void removeSpace(char \*line) {  strtok(line, "\n"); }  Рис. 3 – console.c |
| #ifndef SET\_LAB\_5\_CONSOLE\_H #define SET\_LAB\_5\_CONSOLE\_H   void work();  void removeSpace(char \*);  int console();  #endif //SET\_LAB\_5\_CONSOLE\_H  Рис. 4 – console.h |
| #include <malloc.h> #include <stddef.h> #include <stdio.h> #include <stdlib.h>  #include "SET.h" #include "error.h"  Node \*createnode(int data) {  Node \*newNode = (Node\*) malloc(sizeof(Node));  if (newNode == NULL) {  error(3);  exit(3);  }  if (!newNode) {  return NULL;  }  newNode->data = data;  newNode->next = NULL;  return newNode; }  Set \*makeSet() {  Set \*set = (Set\*) malloc(sizeof(Set));  if (set == NULL) {  error(3);  exit(3);  }  if (!set) {  return NULL;  }  set->head = NULL;  return set; }  void print(Set \*set) {  Node \*current = set->head;  if (set->head == NULL)  return;   for (; current != NULL; current = current->next) {  printf("%d ", current->data);  }  printf("\n"); }  void insert(int data, Set \*set) {  if(find(data, set) == 1) {  Node \*current = NULL;  if (set->head == NULL) {  set->head = createnode(data);  } else {  current = set->head;  while (current->next != NULL) {  current = current->next;  }  current->next = createnode(data);  }  } }  void removeElement(int data, Set \*set) {  Node \*current = set->head;  Node \*previous = current;  while (current != NULL) {  if (current->data == data) {  previous->  next = current->next;  if (current == set->head)  set->  head = current->next;  free(current);  return;  }  previous = current;  current = current->next;  } }   void deleteAll(Set \*set) {  Node \*current = set->head;  Node \*next;  while (current != NULL) {  next = current->next;  free(current);  current = next;  }  free(set); }  int find(int data, Set \*set) {  Node \*current = set->head;  while (current != NULL) {  if (current->data == data) {  return 0;  }  current = current->next;  }  return 1; }  void contains(int data, Set \*set) {  Node \*current = set->head;  if (current == NULL) printf("%d not contains in set\n", data);  while (current != NULL) {  if (current->data == data) {  printf("%d contains in set\n", data);  return;  }  if (current->data != data) {  printf("%d not contains in set\n", data);  }  current = current->next;  } }  **Рис. 5 – SET.c** |
| #ifndef SET\_LAB\_5\_SET\_H #define SET\_LAB\_5\_SET\_H  typedef struct node Node; struct node {  int data;  struct node \*next; };  typedef struct set Set; struct set {  Node \*head; };  Node \*createnode(int data); Set \* makeSet(); void insert(int data, Set \* set); void removeElement(int data, Set \* set); void print(Set \* set); void deleteAll(Set \* set); int find(int data, Set \* set); void contains(int data, Set \* set);  #endif //SET\_LAB\_5\_SET\_H  Рис. 6 – SET.h |
| #include "stdio.h" #include "error.h"    int error(int errorCode) {  switch(errorCode) {  //Ошибка №1: неверный аргумент коммандной строки  case 1: {  printf("Error n.1: Wrong Argument\n");  return 1;  }   //Ошибка №2: ожидалось целое число  case 2: {  printf("Error n.2: Integer expected\n");  return 2;  }   //Ошибка №3:  case 3: {  printf("Error n.3: Error memory\n");  return 3;  }   }  return 0; }  **Рис. 7 – Error.c** |
| #ifndef SET\_LAB\_5\_ERROR\_H #define SET\_LAB\_5\_ERROR\_H    int error(int errorCode);   #endif //SET\_LAB\_5\_ERROR\_H  Рис. 8 – Error.h |

# **5. Ошибки и коды ошибок:**

Однонаправленный set был модифицирован. Чтобы избежать ошибок:

Если в set нет элементов, то removeElement(), deleteAll(), print() – выполняются на пустом set.

Коды:

Ошибка 1: неверное задание аргумента командной строки.

Ошибка 2: ожидалось целое беззнаковое число. (для insert, remove, contains).

Ошибка 3: ошибка выделения памяти.

# **6. Программа и методика тестирования:**

## **6.1 Ошибки**

Тестирование ошибок приведено на рисунках 9 и 10

|  |
| --- |
| Рис. 9 – Ошибка 1 |
| Рис. 10 – Ошибка 2 |

## **6.2 Программа тестирования**

Для проведения тестов создан файл test.c:

|  |
| --- |
| #include "test.h" #include "SET.h" #include "stdio.h"  int doTest() {  Set \*set = makeSet();   for (int i = 0; i < 100; i++) {  add(i, set);  }  print(set);   for (int i = 0; i < 100; i++) {  int bn = find(i, set);  if (bn == 1) {  printf("Element is not in set");  return -1;  }  }   for (int i = 50; i < 100; i++) {  removeElement(i, set);  }  print(set);   for (int i = 0; i < 50; i++) {  int bn = find(i, set);  if (bn == 1) {  printf("Element is not in set");  return -1;  }  }   deleteAll(set);  print(set);  return 0; }  Рис. 11 – test.c |

## **6.3 Рисунки выполнения программы без ошибок:**

|  |
| --- |
| Рис. 12 – Выполнение программы без ошибок |
| Рис. 13 – Выполнение программы без ошибок |

# **7. Вывод:**

В ходе выполнения работы был реализован set, который основан на однонаправленном линейном списке. Создано консольное приложение для обеспечения ввода и вывода информации.