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](#_Toc71656828)

[**2. Метод решения** 3](#_Toc71656829)

[**3. Описание библиотеки:** 3](#_Toc71656830)

[**4. Листинг программы:** 4](#_Toc71656831)

[**4.1 main.c:** 4](#_Toc71656832)

[**4.2 console.c:** 4](#_Toc71656833)

[**4.3 console.h** 5](#_Toc71656834)

[**4.4 SET.c** 6](#_Toc71656835)

[**4.5 SET.h** 7](#_Toc71656836)

[**4.6 error.c** 8](#_Toc71656837)

[**4.7 error.h** 8](#_Toc71656838)

[**5. Коды ошибок:** 9](#_Toc71656839)

[**6. Результат работы программы:** 9](#_Toc71656840)

[**7. Программа тестирования** 10](#_Toc71656841)

[**7.1 Описание тестирования** 10](#_Toc71656842)

[**7.2 Листинг файлов тестирования** 10](#_Toc71656843)

[**7.2.1 main.c** 10](#_Toc71656844)

[**7.2.2 test.c** 11](#_Toc71656845)

[**7.2.3 test.h** 12](#_Toc71656846)

[**7.3 Результаты тестирования** 12](#_Toc71656847)

[**8. Вывод** 12](#_Toc71656848)

# **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) createnode() – создание узла

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

3) insert() – добавление элемента в массив.

4) removeElement() – удаление элемента из set.

5) print() – печать set.

6) deleteAll() – очищение set

7) find() – определение наличия элемента в set

8) contains() – определение наличия элемента в set и вывод сообщения.

# **4. Листинг программы:**

## **4.1 main.c:**

|  |
| --- |
| #include "console.h" #include "test.h" #include <stdio.h> #include <string.h>   int main(int argc, char \*\*argv) {  if (argc <= 1) {  printf("syntax: 0 or 1 (0 for console, 1 for tests)", argv[0]);  return 0;  }  if (!strcmp(argv[1], "0")) {  console();  }  if (!strcmp(argv[1], "1")) {  doTest();  } }  Рис. 2 – main.c |

## **4.2 console.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() {  printf("list of commands:\n");  printf("'insert n' - add n in set (n is integer)\n");  printf("'remove n' - delete n in set (n is integer)\n");  printf("'contains n' - check for availability (n is integer)\n");  printf("'print' - print set\n");  printf("'#' - end work\n\n");  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); }   void removeSpace(char \*line) {  strtok(line, "\n"); }  Рис. 3 – console.c |

## **4.3 console.h**

|  |
| --- |
| #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 |

## **4.4 SET.c**

|  |
| --- |
| #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;  while (current != NULL) {  if (current->data == data) {  printf("%d contains in set\n", data);  return;  }  current = current->next;  }  printf("%d not contains in set\n", data); }  Рис. 5 – SET.c |

## **4.5 SET.h**

|  |
| --- |
| #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 |

## **4.6 error.c**

|  |
| --- |
| #include "stdio.h" #include "error.h"   int error(int errorCode) {  switch(errorCode) {  //Ошибка №1: неверный аргумент   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 |

## **4.7 error.h**

|  |
| --- |
| #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. Коды ошибок:**

Однонаправленный set был модифицирован. Чтобы избежать ошибок: если в set нет элементов, то removeElement(), deleteAll(), print() – выполняются на пустом set.

При запуске программы main.c вызывает метод console, который требует, чтобы пользователь ввёл аргументы. Ошибка 1 и ошибка 2 предназначены для ситуации, когда пользователь вводит неправильное сообщение в консоль.

Ошибка 1: неверное задание метода в консоли

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

Ошибка 3: ошибка выделения памяти.

# **6. Результат работы программы:**

Для запуска основной программы нужно ввести аргумент = 0 в командную строку, тогда main.c вызовет console. Примеры работы представлены ниже:

|  |
| --- |
| Рис. 9 – Неверное задание метода |
| Рис. 10 – Ожидалось целое число |
| Рис. 11 – Результат работы программы (часть 1) |
| Рис. 12 – Результат работы программы (часть 2) |

# **7. Программа тестирования**

## **7.1 Описание тестирования**

Для тестирования программы нужно ввести в командную строку аргумент = 1, тогда main.c вызывает функцию doTest в файле test. Также был создан файл test.c и test.h он включает в себя вызов функций insert(), remove(), contains(), print(), чтобы проверить корректность работы программы.

## **7.2 Листинг файлов тестирования**

### **7.2.1 main.c**

|  |
| --- |
| #include "console.h" #include "test.h" #include <stdio.h> #include <string.h>   int main(int argc, char \*\*argv) {  if (argc <= 1) {  printf("syntax: 0 or 1 (0 for console, 1 for tests)", argv[0]);  return 0;  }  if (!strcmp(argv[1], "0")) {  console();  }  if (!strcmp(argv[1], "1")) {  doTest();  } }  Рис. 13 – main.c |

### **7.2.2 test.c**

|  |
| --- |
| #include "test.h" #include "SET.h" #include "stdio.h"  int doTest() {  Set \*set = makeSet();   for (int i = 0; i < 5; i++) {  printf("insert %d\n", i);  insert(i, set);  }   for (int i = 0; i < 5; i++) {  int bn = find(i, set);  if (bn == 1) {  printf("Add test not passed\n");  return -1;  }  }   printf("Set cantains: ");  print(set);  printf("Test add passed\n\n");   for (int i = 3; i < 5; i++) {  printf("remove %d\n", i);  removeElement(i, set);  }   for (int i = 0; i < 3; i++) {  int bn = find(i, set);  if (bn == 1) {  printf("Remove test not passed\n");  return -1;  }  }   printf("Set cantains: ");  print(set);  printf("Test remove passed\n\n");    printf("Set cantains: ");  print(set);  printf("contains 1\n");  if (find(1, set)) {  printf("Contains test not passed");  return -1;  }  contains(1, set);   printf("contains 3\n");  if (find(3, set) == 0) {  printf("Contains test not passed");  return -1;  }  contains(3,set);  printf("Test contains passed\n");   deleteAll(set);  return 0; }  Рис. 14 – test.c |

### **7.2.3 test.h**

|  |
| --- |
| #ifndef SET\_LAB\_5\_TEST\_H #define SET\_LAB\_5\_TEST\_H  int doTest();  #endif //SET\_LAB\_5\_TEST\_H  Рис. 15 – test.h |

## **7.3 Результаты тестирования**

|  |
| --- |
| Рис. 16 – Результаты тестирования программы |

# **8. Вывод**

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