ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| старший преподаватель |  |  |  | С. А. Рогачев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «ЛИНЕЙНЫЕ И ЦИКЛИЧЕСКИЕ СПИСКИ» |
| по курсу: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |
|  |

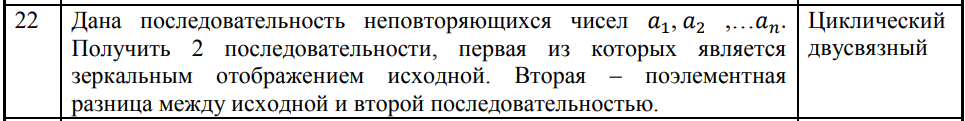
РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4131 |  |  |  | Д. А. Кузнецов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы:** Целью работы является изучение структур данных «линейный список» и «циклический список», а также получение практических навыков их реализации.

**Вариант:** 22



**Листинг программы:**

**int main():**

1. // Memory leaks testing
2. #define \_\_CRTDBG\_MAP\_ALLOC
3. #include <crtdbg.h>
4. #define DEBUG\_NEW new(\_NORMAL\_BLOCK, \_\_FILE\_\_, \_\_LINE\_\_)
5. #define new DEBUG\_NEW
7. #include <iostream>
8. #include "Struct.h"
9. #include <string>;
11. using namespace std;
13. int main()
14. {
15. setlocale(LC\_ALL, "RUS");
17. List\* list = new List();
19. list = fill(list);
21. show(list, "Your origin list: ");

24. List\* reflected = getReflected(list);
25. show(reflected, "Your reflected list: ");
27. List\* difference = getDifference(list, reflected);
28. show(difference, "List of differences: ");
30. cout << endl << "Введите число для удаления из списка: ";
31. int num; cin >> num;
32. list = erase(list, num);
34. show(list, "Result: ");
36. cout << endl << "Введите значение для добавления в список: ";
37. cin >> num;
38. list = add(list, num);
40. show(list, "Result: ");
42. // Clearing memory
43. erase(list);
44. erase(difference);
45. erase(reflected);
47. \_CrtDumpMemoryLeaks();
48. return 0;
49. }

**Struct.cpp:**

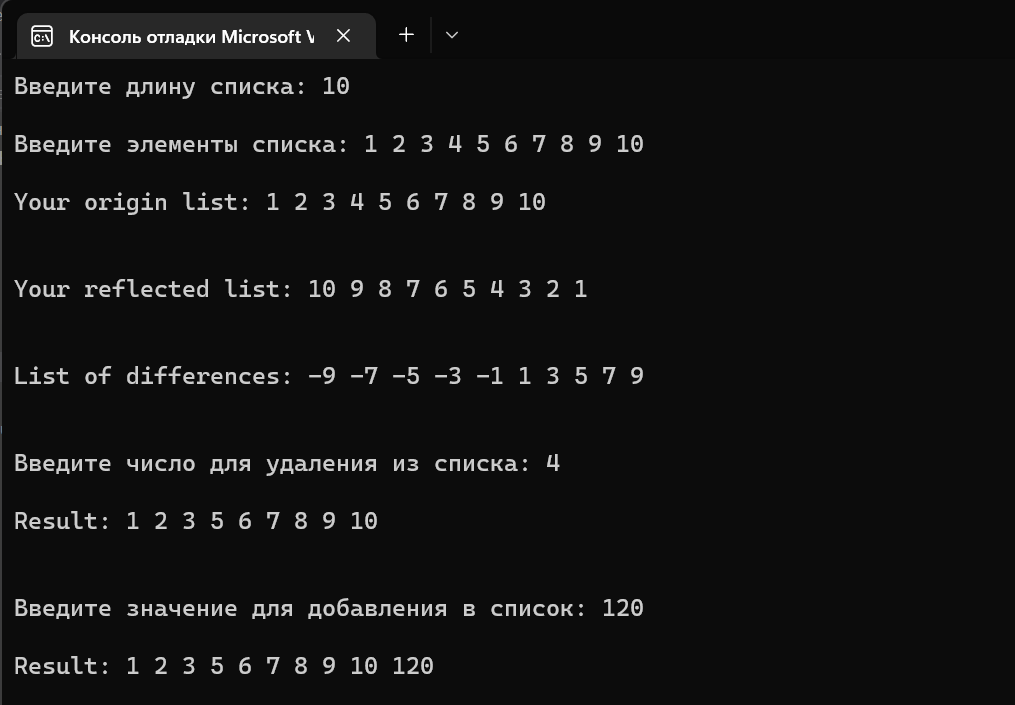
1. #include <iostream>
2. #include "Struct.h"
3. #include <string>
5. using namespace std;
7. List\* add(List\* list, int num) {
8. List\* value = new List();
9. value->data = num;
11. if (list == nullptr) {
12. return value;
13. }
14. // Элемент перед опорным
15. List\* previous = list->previous;
17. // Вставить новый элемент перед опорным и предпоследним
18. previous->next = value;
19. value->previous = previous;
20. value->next = list;
21. list->previous = value;
23. return list;
24. }
26. List\* erase(List\* list, int value) {
28. if (list == nullptr)
29. return list;
31. List\* pivot = list;
32. List\* pointer = list;
34. int k = 0;
36. while (pointer != pivot || k == 0) {
37. if (pointer->data == value) {
39. List\* previous = pointer->previous;
40. List\* next = pointer->next;
42. previous->next = next;
43. next->previous = previous;

46. // Удаление опорного элемента списка
47. if (pointer == list) {
49. if (pointer->next == list) {
50. delete pointer;
51. return nullptr;
52. }
54. delete pointer;
55. return next;
56. }
58. delete pointer;
59. return list;
60. }
62. pointer = pointer->next;
64. k++;
65. }
67. return list;
68. }
70. void erase(List\* list) {
72. if (list == nullptr) {
73. return;
74. }
76. List\* last = list->previous;
78. while (list != last) {
79. List\* temp = list;
80. list = list->next;
82. delete temp;
83. }
85. delete last;
86. }
88. void show(List\* list, string str) {
90. if (list == nullptr) {
91. cout << str << "Your list is empty!" << endl;
92. return;
93. }
95. List\* pivot = list;
96. List\* pointer = list;
98. cout << endl << str;
100. while (pointer != pivot->previous) {
101. List\* previous = pointer->previous;
102. List\* next = pointer->next;
104. // Вывод элемента списка, предыдущего и следующего (для проверки)
105. /\*cout << pointer->data << " " << previous->data << " " << next->data << endl;\*/
107. cout << pointer->data << " ";
109. pointer = pointer->next;
110. }
112. // Последний элемент списка
113. cout << pointer->data;
115. // Для демонстрации работы списка
116. /\*List\* previous = pointer->previous;
117. List\* next = pointer->next;
118. cout << pointer->data << " " << previous->data << " " << next->data << endl;\*/
120. cout << endl << endl;
121. }
123. List\* fill(List\* list) {
125. cout << "Введите длину списка: ";
126. int a; cin >> a;
128. if (a == 0) {
129. return nullptr;
130. }
132. cout << endl << "Введите элементы списка: ";
133. int num;
135. for (int i = 0; i < a; i++) {
136. cin >> num;
138. if (i == 0) {
139. list->data = num;
140. }
141. else {
142. list = add(list, num);
143. }
145. }
147. return list;
148. }
150. List\* getReflected(List\* list) {
152. if (list == nullptr) {
153. return list;
154. }
156. if (list->next == list) {
157. return list;
158. }
160. List\* pointer\_origin = list->previous;
161. List\* pointer\_out = list->previous;
163. List\* out = new List();
165. while (pointer\_origin != list) {
166. if (pointer\_origin == list->previous) {
167. out->data = pointer\_origin->data;
168. }
169. else {
170. add(out, pointer\_origin->data);
171. }
173. pointer\_origin = pointer\_origin->previous;
174. }
176. // Добавление опорного элемента
177. add(out, pointer\_origin->data);
179. return out;
180. }
182. List\* getDifference(List\* list1, List\* list2) {
184. if (list1 == nullptr) {
185. return list1;
186. }
188. List\* out = new List();
190. List\* list1\_pointer = list1;
191. List\* list2\_pointer = list2;
193. out->data = list1\_pointer->data - list2\_pointer->data;
195. list1\_pointer = list1\_pointer->next;
196. list2\_pointer = list2\_pointer->next;
198. while (list1\_pointer != list1) {
199. add(out, list1\_pointer->data - list2\_pointer->data);
201. list1\_pointer = list1\_pointer->next;
202. list2\_pointer = list2\_pointer->next;
203. }
205. return out;
206. }

**Struct.h:**

1. #pragma once
3. #ifndef STRUCT
4. #define STRUCT
6. struct List {
7. int data;
9. List\* previous = this;
10. List\* next = this;
11. };
13. List\* add(List\* list, int num);
14. void show(List\* list, std::string str);
15. List\* fill(List\* list);
16. List\* erase(List\* list, int value);
17. void erase(List\* list);
18. List\* getReflected(List\* list);
19. List\* getDifference(List\* list1, List\* list2);
21. #endif

**Контрольные примеры:**



**Вывод:**

В ходе лабораторной работы я изучил структуру данных «линейный список» и «циклический список», а также получил практические навыки их реализации.