МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

Отчет

Лабораторная работа №2

по дисциплине «Технология разработки программного обеспечения»

**«Разработка оконного приложения»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студенты: | Антонов С.С.  Арнольд Э.В. | Преподаватель: | Гаврилов А. В. |
| Группа: | АММ2-21 | Дата проверки: |  |
| Вариант: | 5 | Балл: |  |
| Дата сдачи: |  |  |  |

Новосибирск 2021

**Постановка задачи**

Разработать оконное приложение, работающее со структурой данных односвязный список.

Функции:

* добавление в конец, получение, вставка и удаление по логическому номеру (индексу). Логический номер – порядковый номер хранимого элемента в соответствии с последовательностью обхода структуры данных (перебора элементов);
* для исходно упорядоченной (например, «двоичное дерево») вставка с сохранением порядка, получение и удаление по по логическому номеру (индексу);
* итератор forEach – для Java: интерфейс обратного вызова с методом void toDo(T v), итератора получает ссылку на интерфейсный объект (анонимный класс), обходит структуру данных и вызывает метод для каждого выбранного элемента. В Scala – формальный параметр = функция;
* сортировка.

**Ход работы**

В основу данного оконного приложения легла структура данных (односвязный список), которая была реализована в первой лабораторной работе.

Для работы с графическим интерфейсом в Kotlin была использована библиотека AndroidX и написана функция App, описывающая элементы интерфейса. Приведём его часть:

1. fun App() {
2. var text by remember { mutableStateOf("") }
3. var outputText by remember { mutableStateOf("0") }
4. var head: MyList<Int>? = null
6. MaterialTheme {
7. Column {
8. Text(outputText)
9. Spacer(modifier = Modifier.height(16.dp))
11. *//Add*
12. TextField(value = text,
13. onValueChange = {text = it},
14. label = { Text("Данные нового элемента") },)
15. Button(onClick = {
16. if(head == null){
17. head = try{
18. MyList(text.toInt())
19. } catch (ex: NumberFormatException){
20. text = ""
21. null
22. return@Button
23. }
24. text = ""
25. outputText = head!!.data.toString()
26. } else {
27. val next = try{
28. MyList(text.toInt())
29. } catch (ex: NumberFormatException){
30. text = ""
31. null
32. return@Button
33. }
34. text = ""
35. head!!.add(next)
36. outputText += " " + next.data.toString()
37. }
38. }) {
39. Text("Add")
40. }

Готовый интерфейс показан на рисунке ниже.

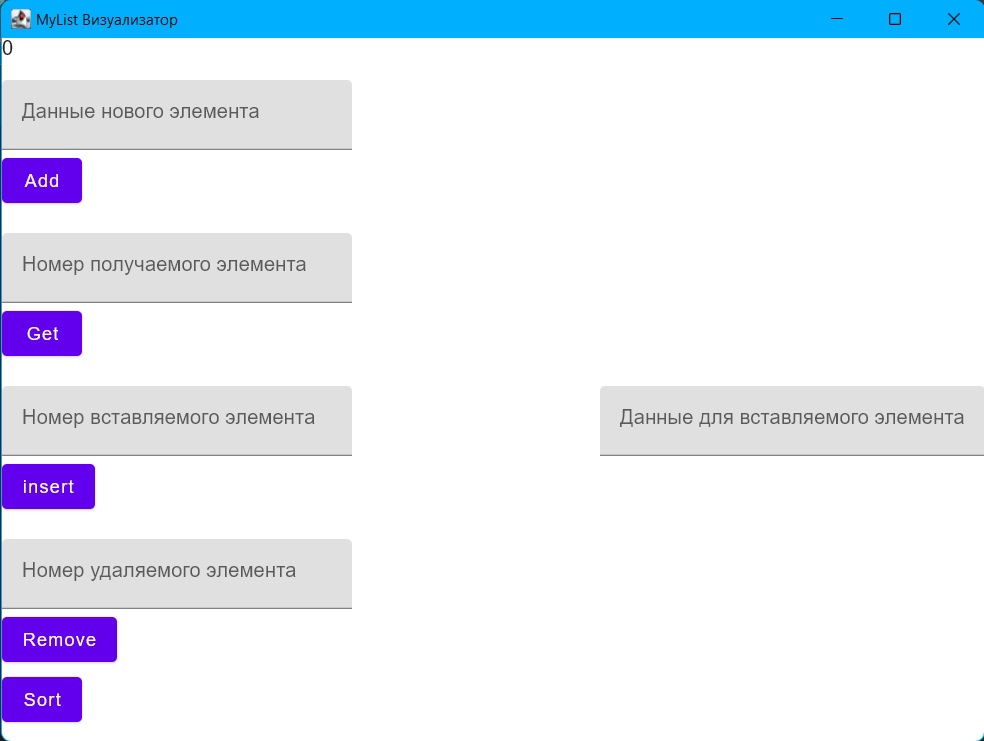


Рисунок 1 – Интерфейс оконного приложения

В левой верхней части интерфейса располагается строка для отображения структуры данных. Ниже расположены текстовые поля и кнопки для взаимодействия со структурой данных.

При загрузке приложения открывается пустая форма. Можно создать новую структуру, добавляя элементы кнопками добавить в конец или на определённый индекс (рисунок 2).

Код метода onClick кнопки Add:

1. Button(onClick = {
2. if(head == null){
3. head = try{
4. MyList(text.toInt())
5. } catch (ex: NumberFormatException){
6. text = ""
7. null
8. return@Button
9. }
10. text = ""
11. outputText = head!!.data.toString()
12. } else {
13. val next = try{
14. MyList(text.toInt())
15. } catch (ex: NumberFormatException){
16. text = ""
17. null
18. return@Button
19. }
20. text = ""
21. head!!.add(next)
22. outputText += " " + next.data.toString()
23. }
24. })

Код метода onClick кнопки Insert, который позволяет вставить элемент на определённую позицию:

1. Button(onClick = {
2. insertStatus = if(head != null){
3. try {
4. head!!.insert(numberToInsert.toInt(), MyList(dataToInsert.toInt(), null))
5. "Status: Завершено!"
6. } catch (ex: NumberFormatException) {
7. "Status: Не завершено!"
8. }
9. } else {
10. head = try {
11. MyList(dataToInsert.toInt(), null)
12. } catch (ex: NumberFormatException) {
13. null
14. }
15. if(head != null){
16. "Status: Завершено!"
17. }
18. else {
19. "Status: Не завершено!"
20. }
21. }
22. if(insertStatus == "Status: Завершено!"){
23. var tmp = head
24. outputText = ""
25. while (tmp != null){
26. outputText += tmp.data.toString() + " "
27. tmp = tmp.next
28. }
29. }
30. })

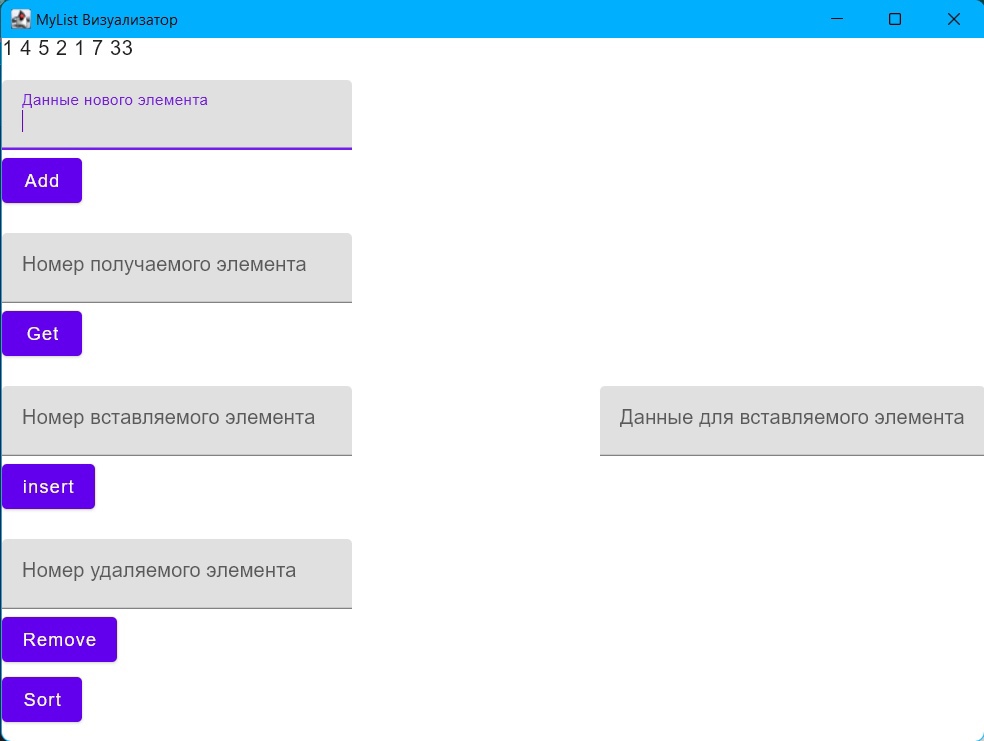


Рисунок 2 – Созданная структура

Для получения значения элемента по его индексу нужно ввести в поле “Номер получаемого элемента” порядковый номер элемента из списка и нажать кнопку Get.

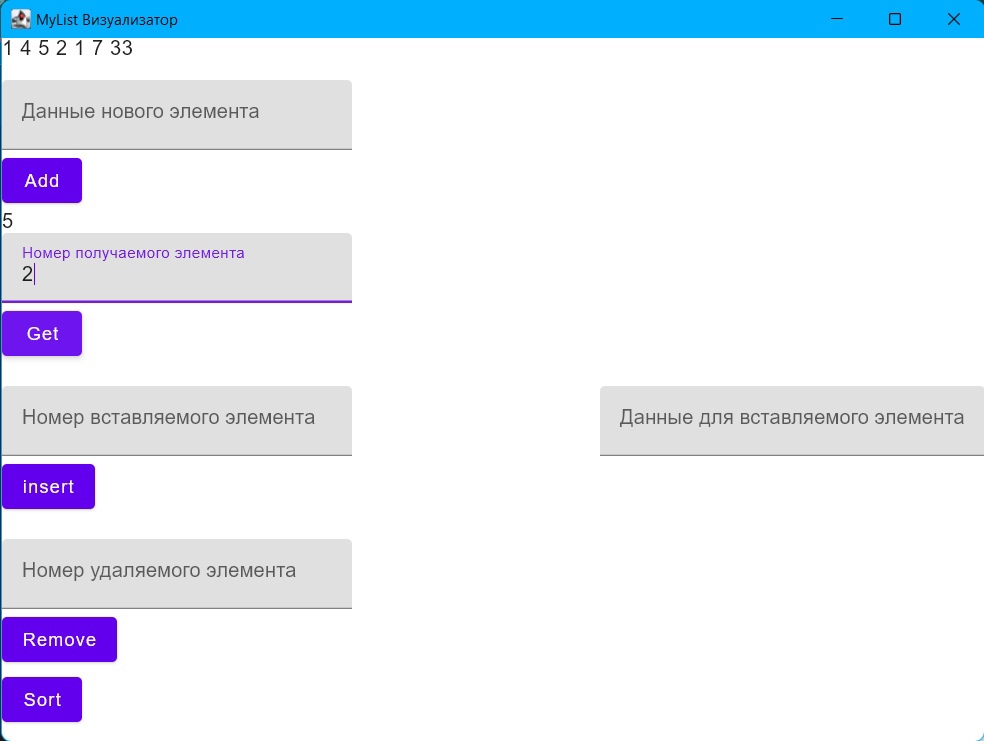


Рисунок 3 – Значение полученного элемента

Для удаления элемента из структуры нужно ввести в поле ID порядковый номер элемента из списка (рисунок 4а и 4б).

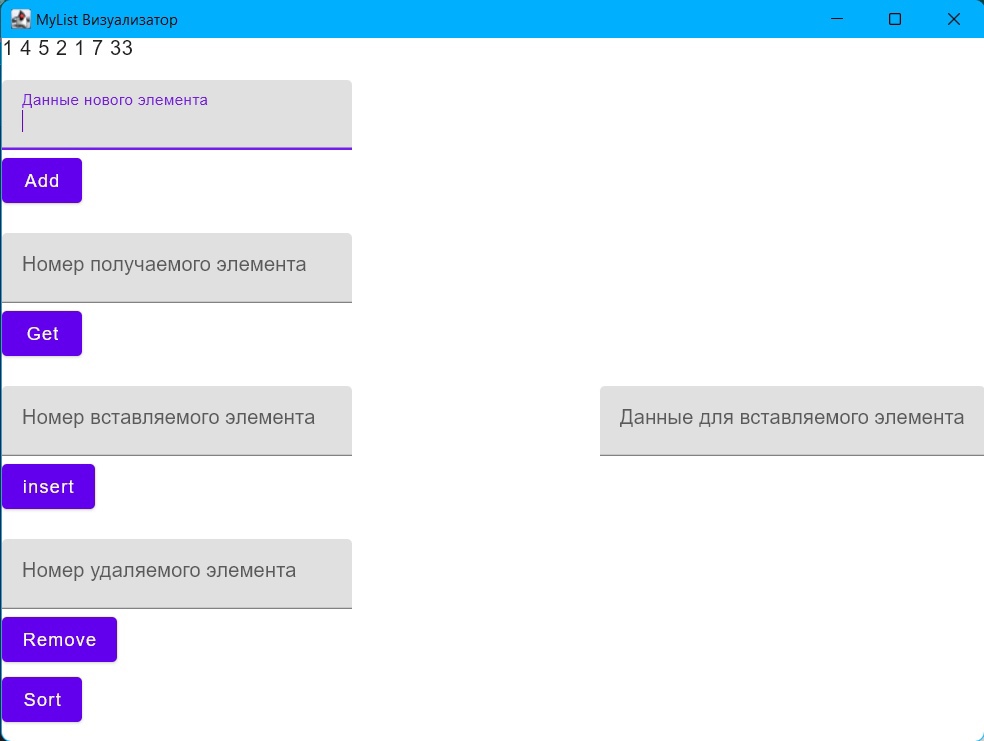


Рисунок 4а – До удаления

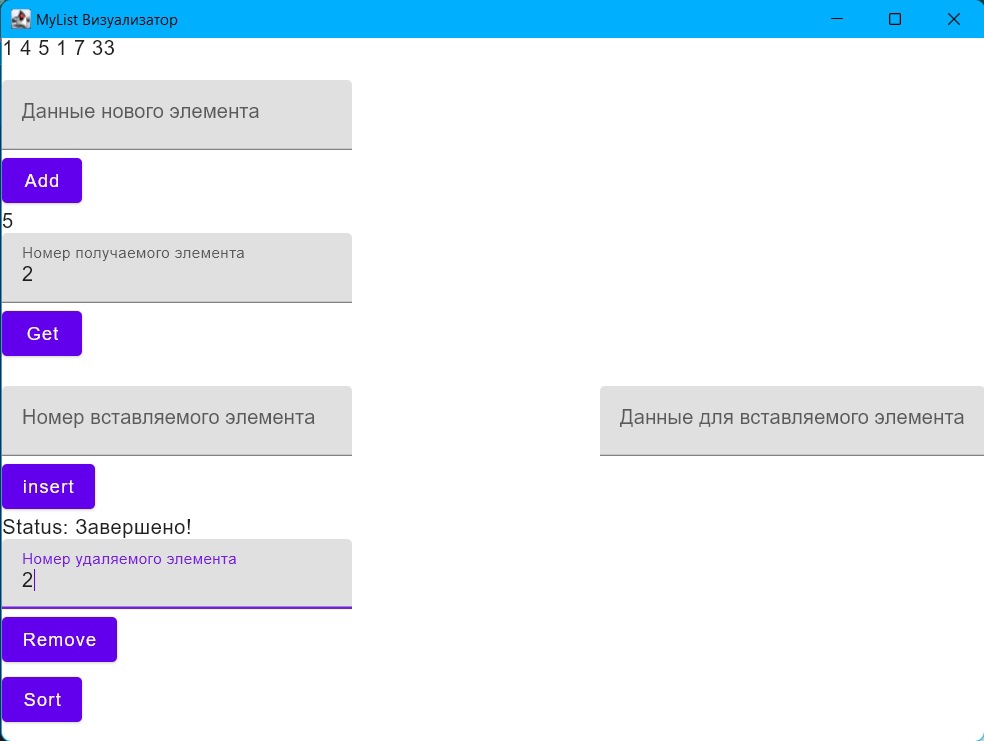


Рисунок 4б – После удаления

Удаление происходит с помощью метода onClick кнопки Remove, который вызывает метод из класса списка remove и передает в него индекс элемента для удаления:

1. Button(onClick = {
2. removeStatus = if(head != null){
3. try {
4. head!!.remove(numberToRemove.toInt())
5. "Status: Завершено!"
6. } catch (ex: NumberFormatException) {
7. "Status: Не завершено!"
8. }
9. } else {
10. "Status: Не завершено!"
11. }
12. if (removeStatus == "Status: Завершено!") {
13. var tmp = head
14. outputText = ""
15. while (tmp != null){
16. outputText += tmp!!.data.toString() + " "
17. tmp = tmp!!.next
18. }
19. }
20. })

По нажатию кнопки «Sort» выполняется сортировка пузырьковым методом. Также нами была устранена ошибка, при которой сортировка не работала, так как не был реализован вывод списка на экран после сортировки.

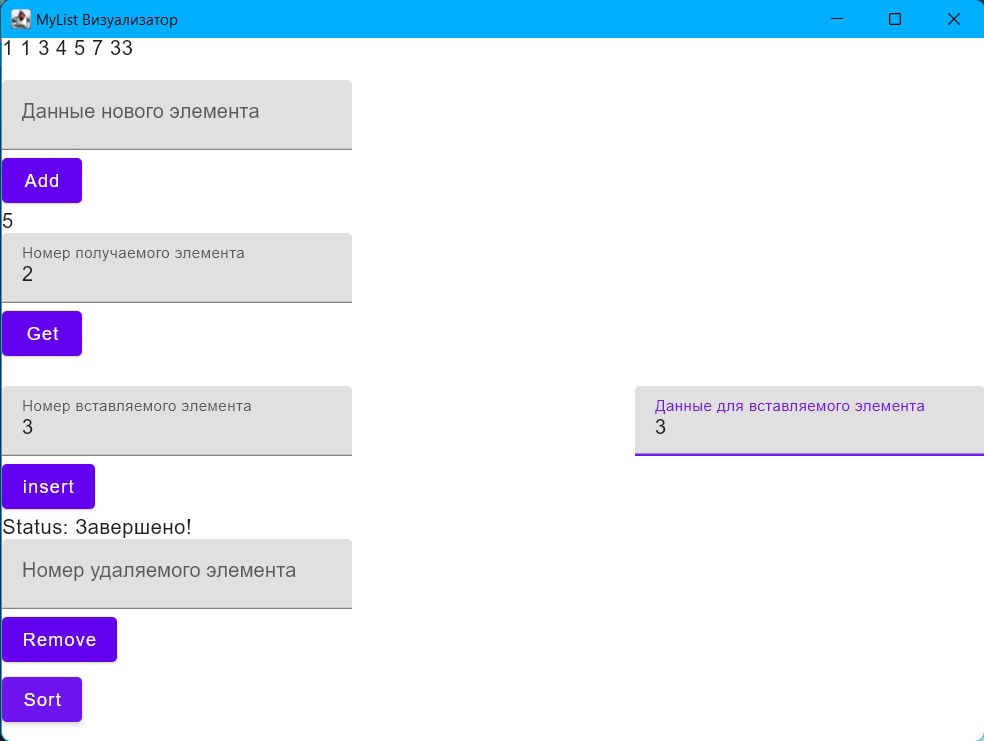


Рисунок 5 – Сортировка элементов

Исправленный метод кнопки «Sort»:

1. Button(onClick = {
2. head?.sort { return@sort it }
3. var tmp = head
4. outputText = ""
5. while (tmp != null){
6. outputText += tmp!!.data.toString() + " "
7. tmp = tmp!!.next
8. }
9. })

Полный код файла Main.kt приведен ниже:

1. import androidx.compose.material.MaterialTheme
2. import androidx.compose.desktop.ui.tooling.preview.Preview
3. import androidx.compose.foundation.layout.Column
4. import androidx.compose.foundation.layout.Row
5. import androidx.compose.foundation.layout.Spacer
6. import androidx.compose.foundation.layout.height
7. import androidx.compose.material.Button
8. import androidx.compose.material.Text
9. import androidx.compose.material.TextField
10. import androidx.compose.runtime.Composable
11. import androidx.compose.runtime.getValue
12. import androidx.compose.runtime.mutableStateOf
13. import androidx.compose.runtime.remember
14. import androidx.compose.runtime.setValue
15. import androidx.compose.ui.Modifier
16. import androidx.compose.ui.unit.dp
17. import androidx.compose.ui.window.Window
18. import androidx.compose.ui.window.application
19. import myList.MyList

22. @Composable
23. @Preview
24. fun App() {
25. var text by remember { mutableStateOf("") }
26. var outputText by remember { mutableStateOf("0") }
27. var head: MyList<Int>? = null
29. MaterialTheme {
30. Column {
31. Text(outputText)
32. Spacer(modifier = Modifier.height(16.dp))
34. *//Add*
35. TextField(value = text,
36. onValueChange = {text = it},
37. label = { Text("Данные нового элемента") },)
38. Button(onClick = {
39. if(head == null){
40. head = try{
41. MyList(text.toInt())
42. } catch (ex: NumberFormatException){
43. text = ""
44. null
45. return@Button
46. }
47. text = ""
48. outputText = head!!.data.toString()
49. } else {
50. val next = try{
51. MyList(text.toInt())
52. } catch (ex: NumberFormatException){
53. text = ""
54. null
55. return@Button
56. }
57. text = ""
58. head!!.add(next)
59. outputText += " " + next.data.toString()
60. }
61. }) {
62. Text("Add")
63. }
65. *//Get*
66. var numberToGet by remember { mutableStateOf("") }
67. var receivedNumber by remember { mutableStateOf("") }
68. Text(receivedNumber)
69. TextField(value = numberToGet,
70. onValueChange = {numberToGet = it},
71. label = { Text("Номер получаемого элемента") },)
72. Button(onClick = {
73. receivedNumber = if(head != null){
74. try {
75. head!!.getBuNumber(numberToGet.toInt())?.data.toString()
76. } catch (ex: NumberFormatException) {
77. "Не верный формат искомого номера"
78. }
79. } else {
80. "Список пуст"
81. }
82. }) {
83. Text("Get")
84. }
86. *//insert*
87. var numberToInsert by remember { mutableStateOf("") }
88. var dataToInsert by remember { mutableStateOf("") }
89. var insertStatus = ""
90. Text(insertStatus)
91. Row {
92. TextField(value = numberToInsert,
93. onValueChange = {numberToInsert = it},
94. label = { Text("Номер вставляемого элемента") },)
95. Spacer(Modifier.weight(1f))
96. TextField(value = dataToInsert,
97. onValueChange = {dataToInsert = it},
98. label = { Text("Данные для вставляемого элемента") })
99. }
100. Button(onClick = {
101. insertStatus = if(head != null){
102. try {
103. head!!.insert(numberToInsert.toInt(), MyList(dataToInsert.toInt(), null))
104. "Status: Завершено!"
105. } catch (ex: NumberFormatException) {
106. "Status: Не завершено!"
107. }
108. } else {
109. head = try {
110. MyList(dataToInsert.toInt(), null)
111. } catch (ex: NumberFormatException) {
112. null
113. }
114. if(head != null){
115. "Status: Завершено!"
116. }
117. else {
118. "Status: Не завершено!"
119. }
120. }
121. if(insertStatus == "Status: Завершено!"){
122. var tmp = head
123. outputText = ""
124. while (tmp != null){
125. outputText += tmp.data.toString() + " "
126. tmp = tmp.next
127. }
128. }
129. }) {
130. Text("insert")
131. }
133. *//remove*
134. var numberToRemove by remember { mutableStateOf("") }
135. var removeStatus by remember { mutableStateOf("") }
136. Text(removeStatus)
137. TextField(value = numberToRemove,
138. onValueChange = {numberToRemove = it},
139. label = { Text("Номер удаляемого элемента") },)
140. Button(onClick = {
141. removeStatus = if(head != null){
142. try {
143. head!!.remove(numberToRemove.toInt())
144. "Status: Завершено!"
145. } catch (ex: NumberFormatException) {
146. "Status: Не завершено!"
147. }
148. } else {
149. "Status: Не завершено!"
150. }
151. if (removeStatus == "Status: Завершено!") {
152. var tmp = head
153. outputText = ""
154. while (tmp != null){
155. outputText += tmp!!.data.toString() + " "
156. tmp = tmp!!.next
157. }
158. }
159. }) {
160. Text("Remove")
161. }
163. *//sort*
164. Button(onClick = {
165. head?.sort { return@sort it }
166. var tmp = head
167. outputText = ""
168. while (tmp != null){
169. outputText += tmp!!.data.toString() + " "
170. tmp = tmp!!.next
171. }
172. }) {
173. Text("Sort")
174. }
175. }
176. }
177. }
179. fun main() = application {
180. Window(title = "MyList Визуализатор", onCloseRequest = ::exitApplication) {
181. App()
182. }
183. }

Полный код файла MyList.kt:

1. package myList
3. class MyList <T> (\_data : T, \_next: MyList<T>? = null) {
4. var data: T = \_data
5. var next: MyList<T>? = \_next
7. fun add(elem: MyList<T>?){
8. var tmp = this
9. while (tmp.next != null) {
10. tmp = tmp.next!!
11. }
12. tmp.next = elem
13. }
14. fun getByData (data: T): MyList<T>?{
15. var tmp: MyList<T>? = this
16. while (tmp != null){
17. if(tmp.data == data){
18. return  tmp
19. }
20. tmp = tmp.next!!
21. }
22. return tmp
23. }
24. fun getBuNumber (number: Int): MyList<T>?{
25. var tmp: MyList<T>? = this
26. var i = 0
27. while (tmp != null){
28. if(i == number){
29. return tmp
30. }
31. tmp = tmp.next!!
32. i++
33. }
34. return tmp
35. }
36. fun insert (number: Int, elem: MyList<T>){
37. var tmp = this
38. var i = 0
39. while (tmp.next != null){
40. if(i == number){
41. break
42. }
43. tmp = tmp.next!!
44. i++
45. }
46. elem.next = tmp.next
47. tmp.next = elem
48. }
49. fun remove(number: Int){
50. var tmp = this
51. var i = 0
52. while (i != number || tmp.next != null){
53. if(i == number) {
54. break
55. }
56. tmp = tmp.next!!
57. i++
58. }
59. if(tmp.next == null){
60. return
61. }
62. tmp.next = (tmp.next)?.next
63. }
64. fun sort(hash: (T) -> Int){
65. var tmp : MyList<T>? = this
66. var n = 0
67. while (tmp != null){
68. tmp = tmp.next
69. n++
70. }
72. for (i in 1 until n){
73. tmp = this
74. for (j in 1 until n){
75. if(tmp?.next != null){
76. if (hash(tmp.data) > hash(tmp.next!!.data)) {
77. val dataTmp = tmp.data
78. tmp.data = tmp.next!!.data
79. tmp.next!!.data = dataTmp
80. }
81. tmp = tmp.next!!
82. }
83. }
84. }
85. }
86. fun foreach(func: (MyList<T>) -> Unit){
87. var tmp: MyList<T>? = this
88. while (tmp != null){
89. func(tmp)
90. tmp = tmp.next
91. }
92. }
93. }

**Заключение**

В ходе выполнения данной лабораторной работы на языке kotlin было реализовано оконное приложения для работы со структурой данных односвязный список. Для структуры были реализованы методы добавления, вставки, удаления, получения значения по индексу, а также метод сортировки списка.