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 №3

по дисциплине «Технология разработки программного обеспечения»

**«Разработка класса структуры данных»**

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

Новосибирск 2021

**Постановка задачи**

Разработать структуру данных «Односвязный список», которая имела бы следующие операции:

* 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. В Scala – формальный параметр = функция.
* Сортировка.

**Ход работы**

Структура данных одномерный массив реализована на языке программирования Scala.

Односвязный список — структура данных, которая состоит из элементов одного типа, связанных между собой последовательно посредством указателей.

В качестве элемента односвязного списка используем «Узел». Данные, которые содержит в себе узел:

* Порядковый номер;
* Name;
* age;
* ссылка на следующий элемент;

Реализованы следующие публичные методы:

* def getData(int key) — получение данных из элемента по ключу.
* def add(String Name, int Age) — добавление нового узла в начало списка.
* def addLast(String Name, int Age) — добавление нового узла в конец списка.
* def printList() — вывод списка на экран.
* def remove(int key) — удаление узла в списке.
* def sort(Comparator comparator) — сортировка списка пузырьковым методом по значению age.
* def eachStep(IForEach forEach) — интерфейс обратного вызова для обхода списка и вывода на печать элементов списка.

Рассмотрим тестовую программу Main. Для проверки работоспособности структуры был создан список. Узлы добавлялись двумя методами:

* В начало списка, метод add.
* В конец списка, метод addLast;

Добавим несколько узлов в список и выведем структуру в консоли с помощью метода eachStep.

1. val list = new PersonList()
2. list.addLast("first", 21)
3. list.addLast("Tom", 11)
4. list.addLast("Стёпа", 17)
5. list.add("Егор", 14)
6. list.addLast("Костя", 22)
7. list.addLast("Катя", 20)
8. list.printList()
10. list.eachStep((list: Person) => {
11. System.out.println(list.Name + ":" + list.age)
12. })

Функции для добавления элементов в начало и конец списка, а также прохода по списку:

1. def add(Name: String, age: Int): Unit = {
2. val tmp = new Person(Name, age)
3. tmp.next = head
4. head = tmp
5. }
6. def addLast(Name: String, age: Int): Unit = {
7. val tmp = new Person(Name, age)
8. var cur = head
9. if (cur == null) {
10. tmp.next = head
11. head = tmp
12. return
13. }
14. while ( {
15. cur.next != null
16. }) cur = cur.next
17. cur.next = tmp
18. tmp.next = null
19. }
20. def eachStep(forEach: IForEach): Unit = {
21. var tmp = head
22. System.out.println("Вывод списка на экран:")
23. while ( {
24. tmp != null
25. }) {
26. forEach.toDo(tmp)
27. tmp = tmp.next
28. }
29. }

Вывод в консоль будет следующим (Рис. 1):

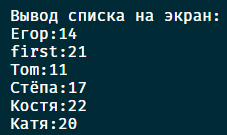


Рис. 1 – Вывод значений узлов списка

Данный вывод говорит нам о том, что методы add и addLast сработали корректно, узлы добавились в соответствующие им позиции.

В настоящий момент наш список не отсортирован. Отсортируем список, используя метод sort, и выведем полученный результат.

1. list.sort(new Ordering[Int]() {
2. override def compare(o1: Int, o2: Int): Int = {
3. val a = o1.asInstanceOf[Integer]
4. val b = o2.asInstanceOf[Integer]
5. if (a < b) 1
6. else if (a == b) 0
7. else -1
8. }
9. })
10. list.printList()
11. list.remove\_At(3)
12. list.printList()

Вывод в консоль (Рис. 2):

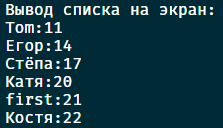


Рисунок 2 – Вывод значений узлов после сортировки списка

Удалим элемент по ключу, вызвав соответствующий метод remove:

list.remove(3)

Код функции remove:

1. def remove(key: Int): Unit = {
2. var cur = head
3. var prev = head
4. if (head != null) {
5. var i = 0
6. while ( {
7. i < getSize
8. }) {
9. if (i == key) if (cur eq head) {
10. head = head.next
11. return
12. }
13. else {
14. prev.next = cur.next
15. return
16. }
17. prev = cur
18. cur = cur.next
20. {
21. i += 1;
22. i - 1
23. }
24. }
25. }
26. }

Теперь выведем список для проверки правильности работы:

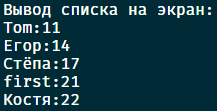


Рисунок 3 – Список после удаления элемента

Листинг кода полностью:

1. import scala.util.control.Breaks.\_
3. object Main {
4. def main(args: Array[String]): Unit = {
5. val list = new PersonList()
6. list.addLast("first", 21)
7. list.addLast("Tom", 11)
8. list.addLast("Стёпа", 17)
9. list.add("Егор", 14)
10. list.addLast("Костя", 22)
11. list.addLast("Катя", 20)
12. list.printList()
14. list.eachStep((list: Person) => {
15. System.out.println(list.Name + ":" + list.age)
16. })
18. list.sort(new Ordering[Int]() {
19. override def compare(o1: Int, o2: Int): Int = {
20. val a = o1.asInstanceOf[Integer]
21. val b = o2.asInstanceOf[Integer]
22. if (a < b) 1
23. else if (a == b) 0
24. else -1
25. }
26. })
27. list.printList()
28. list.remove(3)
29. list.printList()
30. }
31. }
33. class Person{
34. var Name: String = null
35. var age = 0
36. var next: Person = null
38. def this(Name: String, age: Int) = {
39. this()
40. this.Name = Name
41. this.age = age
42. }
44. def getnext: Person = this.next
46. def setNext(Next: Person): Unit = {
47. this.next = Next
48. }
49. def getName: String = this.Name
50. def getAge: Int = this.age
51. }
53. class PersonList {
54. private var head: Person = null
55. private def isEmpty = head == null
57. *//удаление элемента по ключу*
58. def remove(key: Int): Unit = {
59. var cur = head
60. var prev = head
61. if (head != null) {
62. var i = 0
63. while ( {
64. i < getSize
65. }) {
66. if (i == key) if (cur eq head) {
67. head = head.next
68. return
69. }
70. else {
71. prev.next = cur.next
72. return
73. }
74. prev = cur
75. cur = cur.next
77. {
78. i += 1;
79. i - 1
80. }
81. }
82. }
83. }
85. def getSize: Int = {
87. var cur = new Person
88. cur = head
90. var i = 0
91. while (cur != null) {
92. i += 1
93. cur = cur.next
94. }
95. i
96. }
98. def printList(): Unit = {
99. var tmp = new Person
100. tmp = head
102. println("Вывод списка на экран:")
103. while ( {
104. tmp != null
105. }) {
106. println(tmp.Name + ":" + tmp.age)
107. tmp = tmp.next
108. }
109. }
111. *//добавление узла в начало*
112. def add(Name: String, age: Int): Unit = {
113. val tmp = new Person(Name, age)
114. tmp.next = head
115. head = tmp
116. }
118. *//добавление узла в конец*
119. def addLast(Name: String, age: Int): Unit = {
120. val tmp = new Person(Name, age)
121. var cur = head
122. if (cur == null) {
123. tmp.next = head
124. head = tmp
125. return
126. }
127. while ( {
128. cur.next != null
129. }) cur = cur.next
130. cur.next = tmp
131. tmp.next = null
132. }
134. *//Получение данных узла по ключу*
135. def getData(key: Int): Unit = {
136. var cur = head
137. if (head != null) {
138. var i = 0
139. while (i < getSize) {
140. if (i == key) {
141. System.out.println("Извлекаем данные:")
142. System.out.println(cur.Name + ":" + cur.age)
143. System.out.println(cur.getName + ":" + cur.getAge)
144. }
145. cur = cur.next
147. {
148. i += 1;
149. i - 1
150. }
151. }
152. }
153. }
155. def eachStep(forEach: IForEach): Unit = {
157. var tmp = head
158. System.out.println("Вывод списка на экран:")
159. while ( {
160. tmp != null
161. }) {
162. forEach.toDo(tmp)
163. tmp = tmp.next
164. }
165. }
167. *//Сортировка узлов списка по возрастанию (поле age)*
168. def sort(comparator: Ordering[Int]): Unit = {
169. var k: Int = 0
170. var prev: Person = new Person()
171. prev = null
172. var cur: Person = new Person()
173. cur = null
174. var save: Person = new Person()
175. save = null
176. var save2: Person = new Person()
177. save2 = null
178. if (head != null) {
179. var i = 0
180. for (i <- 0 to getSize) {
181. cur = head
182. prev = head
183. breakable { while (cur.next != null) {
184. if (comparator.compare(cur.age, cur.next.age) == -1) {
185. save = cur
186. save2 = cur.next.next
187. if (cur == head) {
188. head = cur.next
189. cur.next.next = save
190. cur.next.next.next = save2
191. }
192. else if (cur.next.next == null) {
193. prev.next = cur.next
194. cur.next.next = save
195. cur.next.next.next = null
196. break
197. }
198. else {
199. prev.next = cur.next
200. cur.next.next = save
201. cur.next.next.next = save2
202. }
203. }
204. prev = cur
205. cur = cur.next
206. }
207. }
208. }
209. }
210. else System.out.println("Список пуст")
211. }
212. }

**Заключение**

В ходе выполнения лабораторной работы нами была реализована структура данных «Односвязный список» на языке программирования Scala. Для структуры были реализованы методы добавления в начало и конец, удаления, поиска, обход и вывод на экран, а также метод сортировки списка пузырьком.