

**Laporan Praktikum**

**Algoritma dan Struktur Data**

Ganjil 2023/2024

Program Studi Teknik Informatika

Institut Teknologi Sumatera

|  |  |
| --- | --- |
| **Modul :** | **Hash** |
| **Nama :** | **Rayhan Fadel Irwanto** |
| **NIM :** | **122140236** |
| **Kelas (Kelas Asal) :** | **ASD RA** |

Instruksi sederhana :

* Disarankan kepada **Praktikan Algoritma Struktur Data** untuk mengeditnya menggunakan Google Docs agar tidak berantakan dan rapi,
* Silahkan mengganti **Nama Modul** baik yang ada pada **Cover** dan **Header** sesuai dengan materi praktikum,
* Gunakan text styling seperti **Heading 1**, **Normal Text** yang telah terformat / Text Style lainnya yang digunakan untuk menjaga estetika laporan,
* Gunakan [**Syntax Highlighter**](https://highlight.hohli.com/index.php)untuk merapikan kode yang sudah Praktikan buat ke dalam Laporan Praktikum.

**Materi Praktikum**

Hash

**Link repl.it Source Code**

<https://replit.com/@RayhanFadel/hash#main.cpp>

**Source Code**

Gunakan [**Syntax Highlighter**](https://highlight.hohli.com/index.php)untuk merapikan Source Code yang dipindahkan dari text editor anda ke dokumen ini.

1. #include <iostream>
2. #include <string>
3. using namespace std;
5. **#define Info(P) (P)->info**
6. #define Next(P) (P)->next
7. #define First(H, i) (H)[i].first
9. typedef string infotype;
10. **const int MaxEl = 26;**
12. typedef struct Tnode \*addressNode;
13. typedef struct Tnode {
14. infotype info;
15. **addressNode next;**
16. } Node;
18. typedef struct THash \*addressHash;
19. typedef struct THash {
20. **addressNode first;**
21. } Hash;
23. void createEmptyHash(addressHash HashTable) {
24. for (int i = 0; i < MaxEl; i++) {
25. **First(HashTable, i) = NULL;**
26. }
27. }
29. addressNode NodeAllocation(infotype x) {
30. **addressNode NewNode;**
32. NewNode = new Node;
33. Info(NewNode) = x;
34. Next(NewNode) = NULL;
36. return NewNode;
37. }
39. bool isEmptyFirst(addressNode First\_Node) { return (First\_Node == NULL); }
41. void NodeDeallocation(addressNode hapus) { delete hapus; }
43. void insertFirst(addressNode \*First\_Node, infotype x) {
44. addressNode NewNode = NodeAllocation(x);
45. **Next(NewNode) = \*First\_Node;**
46. \*First\_Node = NewNode;
47. }
49. void insertLast(addressNode \*First\_Node, infotype x) {
50. **addressNode NewNode = NodeAllocation(x), temp = \*First\_Node;**
52. if (\*First\_Node == NULL) {
53. \*First\_Node = NewNode;
54. } else {
55. **while (Next(temp) != NULL) {**
56. temp = Next(temp);
57. }
59. Next(temp) = NewNode;
60. **}**
61. }
63. void deleteFirst(addressNode \*First\_Node) {
64. if (\*First\_Node != NULL) {
65. **addressNode temp;**
67. temp = \*First\_Node;
68. \*First\_Node = Next(\*First\_Node);
69. Next(temp) = NULL;
71. NodeDeallocation(temp);
72. }
73. }
75. **void deleteAfter(addressNode \*Pred) {**
76. if (\*Pred != NULL && Next(\*Pred) != NULL) {
77. addressNode temp;
79. temp = Next(\*Pred);
80. **Next(\*Pred) = Next(temp);**
82. NodeDeallocation(temp);
83. }
84. }
86. void deleteLast(addressNode \*First\_Node) {
87. if (\*First\_Node != NULL) {
88. addressNode temp, predTemp;
90. **predTemp = NULL;**
91. temp = \*First\_Node;
93. while (Next(temp) != NULL) {
94. predTemp = temp;
95. **temp = Next(temp);**
96. }
98. if (predTemp == NULL) {
99. \*First\_Node = NULL;
100. **} else {**
101. deleteAfter(&predTemp);
102. }
103. }
104. }
106. void insertByModFunc(addressHash HashTable, infotype x) {
107. int index = x[0] - 'a'; // Menggunakan huruf pertama sebagai indeks
108. addressNode \*First\_Node = &First(HashTable, index);
110. **if (isEmptyFirst(\*First\_Node)) {**
111. insertFirst(First\_Node, x);
112. } else {
113. insertLast(First\_Node, x);
114. }
115. **}**
117. void printHashTable(addressHash HashTable) {
118. for (int i = 0; i < MaxEl; i++) {
119. addressNode temp = First(HashTable, i);
120. **cout << "[" << i << "]: ";**
122. while (temp != NULL) {
123. cout << Info(temp) << " -> ";
124. temp = Next(temp);
125. **}**
126. cout << "NULL" << endl;
127. }
128. }
130. **int main() {**
131. addressHash HashTable;
132. HashTable = new Hash[MaxEl];
133. createEmptyHash(HashTable);
135. **insertByModFunc(HashTable, "apel");**
136. insertByModFunc(HashTable, "pisang");
137. insertByModFunc(HashTable, "alpukat");
138. insertByModFunc(HashTable, "duren");
140. **deleteFirst(&First(HashTable, 1));**
142. deleteLast(&First(HashTable, 0));
144. insertFirst(&First(HashTable, 2), "lengkeng");
146. insertLast(&First(HashTable, 0), "anggur");
148. printHashTable(HashTable);
150. delete[] HashTable;
152. return 0;
153. }

**Dokumentasi Hasil Running**

|  |
| --- |
|  |
| **Gambar 1. Output Code Tugas Minggu 6-Hash.**  Program di atas merupakan implementasi struktur data hash table menggunakan linked list dalam bahasa C++. Hash table ini dirancang untuk menyimpan string (infotype) dalam linked list pada setiap bucket. Fungsi-fungsi dasar seperti `createEmptyHash`, `NodeAllocation`, dan `isEmptyFirst` digunakan untuk menginisialisasi hash table, mengalokasikan node baru, dan memeriksa apakah linked list kosong. Operasi-insert (`insertFirst` dan `insertLast`) dan operasi-delete (`deleteFirst`, `deleteAfter`, dan `deleteLast`) diimplementasikan untuk mengelola data dalam hash table. Fungsi `insertByModFunc` menentukan indeks bucket menggunakan huruf pertama dari string dan menyisipkan string ke dalam linked list pada bucket tersebut. Hasil dari operasi ini kemudian dicetak menggunakan fungsi `printHashTable`. Pada contoh ini, string seperti "apel", "pisang", "alpukat", "duren", "lengkeng", dan "anggur" dimasukkan ke dalam hash table, beberapa operasi insert dan delete dilakukan, dan output hasilnya ditampilkan. |

|  |
| --- |
|  |
|  |