

**Laporan Praktikum**

**Algoritma dan Struktur Data**

Ganjil 2023/2024

Program Studi Teknik Informatika

Institut Teknologi Sumatera

|  |  |
| --- | --- |
| **Modul :** | **Hash** |
| **Nama :** | **Muhammad Yusuf** |
| **NIM :** | **122140193** |
| **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/@muhammad1221401/Muhammad-Yusuf122140193ASD-RAHash#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. using namespace std;
4. #define Info(P) (P)->info
5. **#define Next(P) (P)->next**
6. #define First(H, i) (H)[i].first
8. typedef int infotype;
9. const int MaxEl = 26;
11. typedef struct Tnode \*addressNode;
12. typedef struct Tnode {
13. infotype info;
14. addressNode next;
15. **} Node;**
17. typedef struct THash \*addressHash;
18. typedef struct THash {
19. addressNode first;
20. **} Hash;**
22. void createEmptyHash(addressHash HashTable) {
23. for (int i = 0; i < MaxEl; i++) {
24. First(HashTable, i) = NULL;
25. **}**
26. }
28. addressNode NodeAllocation(infotype x) {
29. addressNode NewNode;
31. NewNode = new Node;
32. Info(NewNode) = x;
33. Next(NewNode) = NULL;
35. **return NewNode;**
36. }
38. bool isEmptyFirst(addressNode First\_Node) { return (First\_Node == NULL); }
40. **void NodeDeallocation(addressNode hapus) { delete hapus; }**
42. void insertFirst(addressNode \*First\_Node, infotype x) {
43. addressNode NewNode = NodeAllocation(x);
44. Next(NewNode) = \*First\_Node;
45. **\*First\_Node = NewNode;**
46. }
48. void insertLast(addressNode \*First\_Node, infotype x) {
49. addressNode NewNode = NodeAllocation(x), temp = \*First\_Node;
51. if (\*First\_Node == NULL) {
52. \*First\_Node = NewNode;
53. } else {
54. while (Next(temp) != NULL) {
55. **temp = Next(temp);**
56. }
58. Next(temp) = NewNode;
59. }
60. **}**
62. void deleteFirst(addressNode \*First\_Node) {
63. if (\*First\_Node != NULL) {
64. addressNode temp;
66. temp = \*First\_Node;
67. \*First\_Node = Next(\*First\_Node);
68. Next(temp) = NULL;
70. **NodeDeallocation(temp);**
71. }
72. }
74. void deleteAfter(addressNode \*Pred) {
75. **if (\*Pred != NULL && Next(\*Pred) != NULL) {**
76. addressNode temp;
78. temp = Next(\*Pred);
79. Next(\*Pred) = Next(temp);
81. NodeDeallocation(temp);
82. }
83. }
85. **void deleteLast(addressNode \*First\_Node) {**
86. if (\*First\_Node != NULL) {
87. addressNode temp, predTemp;
89. predTemp = NULL;
90. **temp = \*First\_Node;**
92. while (Next(temp) != NULL) {
93. predTemp = temp;
94. temp = Next(temp);
95. **}**
97. if (predTemp == NULL) {
98. \*First\_Node = NULL;
99. } else {
100. **deleteAfter(&predTemp);**
101. }
102. }
103. }
105. **void insertByModFunc(addressHash HashTable, infotype x) {**
106. int index = x % MaxEl;
107. addressNode \*First\_Node = &First(HashTable, index);
109. if (isEmptyFirst(\*First\_Node)) {
110. **insertFirst(First\_Node, x);**
111. } else {
112. insertLast(First\_Node, x);
113. }
114. }
116. void printHashTable(addressHash HashTable) {
117. for (int i = 0; i < MaxEl; i++) {
118. addressNode temp = First(HashTable, i);
119. cout << "[" << i << "]: ";
121. while (temp != NULL) {
122. cout << Info(temp) << " -> ";
123. temp = Next(temp);
124. }
125. **cout << "NULL" << endl;**
126. }
127. }
129. int main() {
130. **addressHash HashTable;**
131. HashTable = new Hash[MaxEl];
132. createEmptyHash(HashTable);
134. insertByModFunc(HashTable, 15);
135. **insertByModFunc(HashTable, 27);**
136. insertByModFunc(HashTable, 42);
138. deleteFirst(&First(HashTable, 1));
140. **deleteLast(&First(HashTable, 0));**
142. insertFirst(&First(HashTable, 2), 8);
144. insertLast(&First(HashTable, 0), 33);
146. printHashTable(HashTable);
148. // hindari kebocoran memori
149. delete[] HashTable;
151. return 0;
152. }

**Dokumentasi Hasil Running**

|  |
| --- |
|  |
| **Gambar 1.** Output Code Hash Latihan Minggu 6.  Program di atas adalah implementasi hash table dalam C++. Setiap bucket dalam hash table adalah linked list, dan setiap node dalam linked list memiliki informasi (info) dan pointer ke node selanjutnya (next). Terdapat beberapa fungsi dasar untuk mengelola hash table, seperti `createEmptyHash` untuk membuat hash table kosong, `NodeAllocation` untuk mengalokasikan node baru, `isEmptyFirst` untuk memeriksa apakah linked list kosong, dan fungsi-fungsi lainnya untuk menyisipkan, menghapus, dan mencetak elemen dalam hash table.  Fungsi `insertByModFunc` memasukkan elemen ke dalam hash table berdasarkan hasil modulus dari nilai elemen tersebut. Jika bucket kosong, elemen dimasukkan di awal linked list; jika tidak, elemen dimasukkan di akhir linked list. Setelah beberapa operasi insert dan delete, hasil dari hash table dicetak menggunakan fungsi `printHashTable`. Dalam contoh ini, angka-angka seperti 15, 27, 42, 8, dan 33 dimasukkan ke dalam hash table, dan beberapa operasi insert dan delete dilakukan untuk menghasilkan output yang ditampilkan pada Gambar 1. |

|  |
| --- |
|  |
|  |