Tugas Praktikum Algoritma dan Struktur Data

Nama : Agil Deriansyah Hasan
Nim : 4522210125

Dosen Pengajar :

Dra.SRI REZEKI CANDRA NURSARI,M.Kom
Prak. Algoritma dan Struktur Data - I

S1-Teknik Informatika
Fakultas Teknik
Universitas Pancasila 2023/2024

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

```cpp
#include <iostream>
#include <cstring>
#include <cstdlib>
#define hashsize 5

using namespace std;

typedef struct _node {
    char *nama;
    char *desc;
    struct _node *next;
} node;

static node *hashtab[hashsize];

void inithashtab() {
    for (int i = 0; i < hashsize; i++) {
        hashtab[i] = NULL;
    }
}

unsigned int hashfunc(char *s) {
    unsigned int h = 0;
    while (*s) {
        h = *s + h * 31;
        s++;
    }
    return h % hashsize;
}

node* lookup(char *n) {
    unsigned int hi = hashfunc(n);
    node* np = hashtab[hi];
    while (np != NULL) {
        if (strcmp(np->nama, n) == 0) {
            return np;
        }
        np = np->next;
    }
    return NULL;
}

char* m_strdup(const char *o) {
    int l = strlen(o) + 1;
    char *ns = (char*)malloc(l * sizeof(char));
    if (ns != NULL) {
        strcpy(ns, o);
    }
    return ns;
}
```

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

F:\cnthprak13.cpp

```cpp
char* get(char* nama) {
    node* n = lookup(nama);
    if (n == NULL) {
        return NULL;
    } else {
        return n->desc;
    }
}

int install(char* nama, char* desc) {
    unsigned int hi;
    node* np;
    if ((np = lookup(nama)) == NULL) {
        hi = hashfunc(nama);
        np = (node*)malloc(sizeof(node));
        if (np == NULL) {
            return 0;
        }
        np->nama = m_strdup(nama);
        if (np->nama == NULL) return 0;
        np->desc = NULL;
        np->next = hashtab[hi];
        hashtab[hi] = np;
    } else {
        free(np->desc);
    }
    np->desc = m_strdup(desc);
    if (np->desc == NULL) return 0;
    return 1;
}

void displaytable() {
    for (int i = 0; i < hashsize; i++) {
        if (hashtab[i] != NULL) {
            node* t = hashtab[i];
            while (t != NULL) {
                cout << t->nama << ": " << t->desc << endl;
                t = t->next;
            }
        }
    }
}
```

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

```cpp
    }
}

void cleanup() {
    for (int i = 0; i < hashsize; i++) {
        node* np = hashtab[i];
        while (np != NULL) {
            node* t = np->next;
            free(np->nama);
            free(np->desc);
            free(np);
            np = t;
        }
        hashtab[i] = NULL;
    }
}

void data() {
    const char* nama[] = {"nama", "alamat", "telepon", "cita-cita", "sekolah"};
    const char* descs[] = {"ichan", "Depok", "08999999991", "Dosen", "Mahasiswa"};
    inithashtab();
    for (int i = 0; i < 5; i++) {
        install((char*)nama[i], (char*)descs[i]);
    }
    install((char*)"telpon", (char*)"9999999");
    cout << "Hasilnya adalah:" << endl;
    for (int i = 0; i < 5; i++) {
        cout << nama[i] << ": " << get((char*)nama[i]) << endl;
    }
    cout << "telpon: " << get((char*)"telpon") << endl;
}

```

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

```cpp
    }
    cout << "telpon: " << get((char*)"telpon") << endl;
}

int main() {
    int pilihan;
    do {
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++" << endl;
        cout << "| Nama || Alamat || Telpon || Cita-Cita || Sekolah |" << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++" << endl;
        cout << "~~~~~~~~~~~~~~~~" << endl;
        cout << "| Menu Pilihan |" << endl;
        cout << "~~~~~~~~~~~~~~~~" << endl;
        cout << "1. Tampilan Data" << endl;
        cout << "2. Display Data" << endl;
        cout << "3. CleanUp" << endl;
        cout << "4. Exit" << endl;
        cout << "| Pilihan Anda? |" << endl;
        cin >> pilihan;
        switch (pilihan) {
            case 1:
                data();
                break;
            case 2:
                displaytable();
                break;
            case 3:
                cleanup();
                cout << "Data telah dibersihkan." << endl;
                break;
            case 4:
                cleanup();
                cout << "Terima Kasih! " << endl;
                break;
            default:
                cout << "Pilihan tidak valid." << endl;
        }
    } while (pilihan != 4);
    return 0;
}

```

```
F:\>g++ pasd13-1.cpp -o 1

F:\>1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
| Nama || Alamat || Telpon || Cita-Cita || Sekolah |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
~~~~~~~~~~~~~~~~
| Menu Pilihan |
~~~~~~~~~~~~~~~~
1. Tampilan Data
2. Display Data
3. CleanUp
4. Exit
| Pilihan Anda? |
1
Hasilnya adalah:
nama: ichan
alamat: Depok
telepon: 08999999991
cita-cita: Dosen
sekolah: Mahasiswa
telpon: 9999999
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
| Nama || Alamat || Telpon || Cita-Cita || Sekolah |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
~~~~~~~~~~~~~~~~
| Menu Pilihan |
~~~~~~~~~~~~~~~~
1. Tampilan Data
2. Display Data
3. CleanUp
4. Exit
| Pilihan Anda? |
3
Data telah dibersihkan.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
| Nama || Alamat || Telpon || Cita-Cita || Sekolah |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
~~~~~~~~~~~~~~~~
| Menu Pilihan |
~~~~~~~~~~~~~~~~
1. Tampilan Data
2. Display Data
3. CleanUp
4. Exit
| Pilihan Anda? |
2
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
| Nama || Alamat || Telpon || Cita-Cita || Sekolah |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~++++++++++++++
~~~~~~~~~~~~~~~~
| Menu Pilihan |
~~~~~~~~~~~~~~~~
1. Tampilan Data
2. Display Data
```

Pseudocode :

```
Kamus/Deklarasi Variabel fungsi inithashtab
i = int

Algoritma/Deskripsi fungsi inithashtab
 for (int i = 0; i < hashsize; i++)
       hashtab[i] = NULL
endfor

Kamus/Deklarasi Variabel fungsi hashfunc
*s = char
h = int

Algoritma/Deskripsi fungsi hashfunc(*s)
h=0
while (*s)
       h = *s + h * 31
       s++
endwhile
print h % hashsize
```

```
Kamus/Deklarasi Variabel fungsi lookup
*n = int
hi = int

Algoritma/Deskripsi fungsi lookup(*n)
hi = hashfunc(n)
   node* np = hashtab[hi]
   while (np != NULL)
      if (strcmp(np->nama, n) == 0)
         return np
      endif
      np = np->next
    endwhile
print NULL

Kamus/Deklarasi Variabel fungsi m_strdup
*o,*ns = char
l = int

Algoritma/Deskripsi fungsi m_strdup(*o)
l = strlen(o) + 1
   char *ns = (char*)malloc(l * sizeof(char))
   if (ns != NULL)
      strcpy(ns, o)
   endif
   print ns
```

```
Kamus/Deklarasi Variabel fungsi get
nama = char

Algoritma/Deskripsi fungsi get(nama)
node* n = lookup(nama)
   if (n == NULL)
      print NULL
   else
      print  n->desc
  endif

Kamus/Deklarasi Variabel fungsi install
nama,desc = int
hi = unsigned int

Algoritma/Deskripsi fungsi install(nama,desc)
node* np
   if ((np = lookup(nama)) == NULL)
      hi = hashfunc(nama)
      np = (node*)malloc(sizeof(node))
      if (np == NULL)
         print 0
         endif
      np->nama = m_strdup(nama)
      if (np->nama == NULL) print 0
      np->desc = NULL
      np->next = hashtab[hi]
      hashtab[hi] = np
    else
      free(np->desc)
  endif
   np->desc = m_strdup(desc)
   if (np->desc == NULL) print 0
   print 1
```

```
Kamus/Deklarasi Variabel fungsi displaytable
i = int

Algoritma/Deskripsi fungsi displaytable
for (int i = 0; i < hashsize; i++)
       if (hashtab[i] != NULL)
          node* t = hashtab[i]
          while (t != NULL)
             print t->nama;t->desc
             t = t->next
         endwhile
        endif
endfor

Kamus/Deklarasi Variabel fungsi cleanup
i = int

Algoritma/Deskripsi fungsi cleanup
for (int i = 0; i < hashsize; i++)
       node* np = hashtab[i]
       while (np != NULL)
          node* t = np->next
          free(np->nama)
          free(np->desc)
          free(np)
          np = t
         endwhile
       hashtab[i] = NULL
endfor
```

```
Kamus/Deklarasi Variabel fungsi data
nama[],descs[] = const char
i = int

Algoritma/Deskripsi fungsi data
nama[] = {"nama", "alamat", "telepon",
"cita-cita", "sekolah"}
descs[] = {"ichan", "Depok", "08999999991",
"Dosen", "Mahasiswa"}
   inithashtab()
   for (int i = 0; i < 5; i++)
      install((char*)nama[i], (char*)descs[i])
   endfor
   install((char*)"telpon", (char*)"9999999")
   for (int i = 0; i < 5; i++)
      print nama[i];get((char*)nama[i])
   endfor
   print get((char*)"telpon")
```

```
Kamus/Deklarasi Variabel fungsi utama
pilihan = int

Algoritma/Deskripsi fungsi utama
typedef struct _node{ *nama, *desc, *next}
_node node
do
      input pilihan
      switch (pilihan)
         case 1:
            data()
            break
         case 2:
            displaytable()
            break
         case 3:
            cleanup()
            print "Data telah dibersihkan."
            break
         case 4:
            cleanup()
            print "Terima Kasih! "
            break
         default:
            print "Pilihan tidak valid."
        endswitch
enddo
        while (pilihan != 4);
  print  0
```

Algoritma :

1. Membuat fungsi inithashtab
2. Selama (i=0) maka kerjakan baris 3 s.d 4
3. hashtab[i]=NULL
4. i++
5. Membuat fungsi hashfunc(*s)
6. h=0
7. Selama(*s) maka kerjakan baris 8 s.d 9
8. h=*s+h*31
9. s++
10. Mencetak h%hashsize
11. Membuat fungsi lookup(*n)
12. hi=hashfunc(n)
13. node* np=hashtab[hi]
14. Selama (np != NULL) maka kerjakan baris 15 s.d 17
15. Jika (strcmp(np->nama, n) == 0) maka kerjakan baris 16
16. Mencetak np
17. np = np->next
18. Mencetak NULL
19. Membuat fungsi m_strdup(*o)
20. l = strlen(o)+1
21. *ns = (char*)malloc(l * sizeof(char))
22. Jika (ns != NULL) maka kerjakn baris 23
23. strcpy(ns, o)
24. Mencetak ns
25. Membuat fungsi get(nama)
26. node* n = lookup(nama))
27. Jika (n == NULL) maka kerjakan baris 28
28. kalau tidak baris 29
29. Mencetak NULL
30. Mencetak n -> desc
31. Membuat fungsi install(nama,desc)
32. node* np
33. Jika ((np = lookup(nama)) == NULL) maka kerjakan baris 34 s.d 43 kalau tidak baris 44
34. hi = hashfunc(nama)
35. np = (node*)malloc(sizeof(node))
36. Jika (np == NULL) maka kerjakan baris 37
37. Mencetak 0
38. np->nama = m_strdup(nama)
39. Jika (np->nama == NULL)
40. Mencetak 0
41. np->desc = NULL
42. np->next = hashtab[hi]
43. hashtab[hi] = np
44. free(np->desc)
45. np->desc = m_strdup(desc)
46. Jika (np->desc == NULL)
47. Mencetak 0
48. Mencetak 1
49. Membuat fungsi displaytable
50. Selama (i=0) maka kerjakan baris 51 s/d 57
51. Jika (hashtab[i] != NULL) maka kerjakan baris 52 s.d 56
52. node* t=hashtab[i]

53. Selama (t!=NULL) maka kerjakan baris 54 s.d 56
54. Mencetak/Menampilkan Nilai t->nama
55. Mencetak/Menampilkan Nilai t->desc
56. t=t->next
57. i++
58. Membuat fungsi cleanup
59. Selama (i=0) maka kerjakan baris 60 s.d 68
60. node* np = hashtab[i]
61. Selama (np != NULL) maka kerjakan baris 62 s.d 66
62. node* t = np->next
63. free(np->nama)
64. free(np->desc)
65. free(np)
66. np = t
67. hashtab[i] = NULL
68. i++
69. Membuat fungsi data
70. nama[] = {"nama", "alamat", "telepon", "cita-cita", "sekolah"};
71. descs[] = {"ichan", "Depok", "08999999991", "Dosen", "Mahasiswa"};
72. Memanggil fungsi inithashtab()
73. Selama (i=0) maka kerjakan baris 73 s.d 75
74. install((char*)nama[i], (char*)descs[i])
75. i++
76. install((char*)"telpon", (char*)"9999999")
77. Selama (int i = 0) maka kerjakan baris 77 s.d 79
78. Mencetak nama[i] ; get((char*)nama[i])
79. i++
80. get((char*)"telpon")

81. Membuat fungsi utama
82. Deklarasi struktur (typedef struct _node {*nama,*desc,*next})
83. Mendefinisikan struktur node (_node node)
84. do
85. Menginput/Memasukkan Nilai pilihan
86. switch(pilihan
87. case 1:
88. Memanggil fungsi data();
89. break
90. case 2:
91. Memanggil fungsi displaytable()
92. break
93. case 3:
94. Memanggil fungsi cleanup()
95. Mencetak "Data telah dibersihkan."
96. break
97. case 4:
98. Memanggil fungsi cleanup()
99. Mencetak "Terima Kasih! "
100. break
101. default:
102. Mencetak "Pilihan tidak valid."
103. Selama (pilihan != 4)
104. print 0

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

```cpp
#include <iostream>
#include <cstdlib>

using namespace std;

struct data {
    int key;
    int value;
};

struct data *array;
int capacity = 10;
int size = 0;

int hashcode(int key) {
    return key % capacity;
}

int if_prime(int n) {
    if (n <= 1) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int get_prime(int n) {
    if (n % 2 == 0) n++;
    while (!if_prime(n)) n += 2;
    return n;
}

void init_array() {
    capacity = get_prime(capacity);
    array = (struct data*)malloc(capacity * sizeof(struct data));
    for (int i = 0; i < capacity; i++) {
        array[i].key = 0;
        array[i].value = 0;
    }
}

void insert(int key) {
    int index = hashcode(key);
    if (array[index].value == 0) {
        array[index].key = key;
        array[index].value = 1;
        size++;
        cout << "Kunci " << key << " telah di-insert." << endl;
    } else if (array[index].key == key) {
        array[index].value += 1;
        cout << "Kunci " << key << " telah di-update." << endl;
    } else {
        cout << "Elemen tidak dapat di-insert." << endl;
    }
}

```

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

```cpp

int main() {
    int choice, key;
    init_array();
    do {
        cout << "Implementasi Penggunaan Hash Table" << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
        cout << "           M E N U                " << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
        cout << " 1. Sisip item pada hash table " << endl;
        cout << " 2. Hapus item pada hash table " << endl;
        cout << " 3. Cek Ukuran pada hash table " << endl;
        cout << " 4. Menampilkan hash table " << endl;
        cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;
        cout << "Pilihan Anda: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Sisip item pada hash table" << endl;
                cout << "Masukkan kunci: ";
                cin >> key;
                insert(key);
                break;
            case 2:
                cout << "Hapus item pada hash table" << endl;
                cout << "Masukkan kunci: ";
                cin >> key;
                remove_element(key);
                break;
            case 3:
                cout << "Ukuran pada hash table: " << size_of_hashtable() << endl;
                break;
            case 4:
                display();
                break;
            default:
                cout << "Salah input." << endl;
        }
        cout << "Melanjutkan (tekan 1 apabila ya): ";
        cin >> choice;
    } while (choice == 1);

    free(array);
    return 0;
}

```

cnthprak13.cpp | pasd13-1.cpp | pasd13-2.cpp | tgsbsr.cpp

```cpp
void remove_element(int key) {
    int index = hashcode(key);
    if (array[index].value == 0) {
        cout << "Kunci tidak ada." << endl;
    } else {
        array[index].key = 0;
        array[index].value = 0;
        size--;
        cout << "Kunci " << key << " telah dihapus." << endl;
    }
}

void display() {
    for (int i = 0; i < capacity; i++) {
        if (array[i].value == 0) {
            cout << "Array [" << i << "] tidak ada elemen." << endl;
        } else {
            cout << "Array [" << i << "] mempunyai elemen kunci " << array[i].key << " dan nilai " << array[i].value << endl;
        }
    }
}

int size_of_hashtable() {
    return size;
}
```

```
F:\>g++ pasd13-2.cpp -o 1

F:\>1
Implementasi Penggunaan Hash Table
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          M E N U
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 1. Sisip item pada hash table
 2. Hapus item pada hash table
 3. Cek Ukuran pada hash table
 4. Menampilkan hash table
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Pilihan Anda: 1
Sisip item pada hash table
Masukkan kunci: 5
Kunci 5 telah di-insert.
Melanjutkan (tekan 1 apabila ya): 1
Implementasi Penggunaan Hash Table
```

```
Command Prompt - 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          M E N U
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 1. Sisip item pada hash table
 2. Hapus item pada hash table
 3. Cek Ukuran pada hash table
 4. Menampilkan hash table
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Pilihan Anda: 4
Array [0] tidak ada elemen.
Array [1] tidak ada elemen.
Array [2] tidak ada elemen.
Array [3] tidak ada elemen.
Array [4] tidak ada elemen.
Array [5] mempunyai elemen kunci 5 dan nilai 1
Array [6] tidak ada elemen.
Array [7] tidak ada elemen.
Array [8] tidak ada elemen.
Array [9] tidak ada elemen.
Array [10] tidak ada elemen.
Melanjutkan (tekan 1 apabila ya):
```

Pseudocode :

```
Kamus/Deklarasi Variabel fungsi hashcode
key = int

Algoritma/Deskripsi fungsi hashcode(key)
Print key % capacity

Kamus/Deklarasi Variabel fungsi if_prime
n,i = int

Algoritma/Deskripsi fungsi if_prime(n)
 if (n <= 1) print 0
   for (int i = 2; i * i <= n; i++)
      if (n % i == 0) print 0
   endfor
   print 1

Kamus/Deklarasi Variabel fungsi get_prime
n = int

Algoritma/Deskirpsi fungsi get_prime(n)
if (n % 2 == 0) n++
   while (!if_prime(n)) n += 2
   print n
```

```
Kamus/Dekalrasi Variabel fungsi init_array
i = int

Algoritma/Deskripsi fungsi init_array
capacity = get_prime(capacity)
array = (struct data*)malloc(capacity*sizeof(struct
data))
   for (int i = 0; i < capacity; i++)
      array[i].key = 0;
      array[i].value = 0
   endfor

Kamus/Deklarasi Variabel fungsi insert
key, index = int

Algoritma/Deskripsi fungsi insert(key)
index = hashcode(key);
   if (array[index].value == 0)
      array[index].key = key
      array[index].value = 1
      size++
      print key
    else if (array[index].key == key)
      array[index].value += 1
      print key
   else
      print "Elemen tidak dapat di-insert."
   endif
```

```
Kamus/Deklarasi Variabel fungsi remove_element
key,index = key

Algoritma/Deskripsi fungsi remove_element(key)
index = hashcode(key)
   if (array[index].value == 0)
      print "Kunci tidak ada."
    else
      array[index].key = 0
      array[index].value = 0
      size--
      print key
   endif

Kamus/Deklarasi Variabel fungsi display
i = int

Algoritma/Deskripsi fungsi display
for (int i = 0; i < capacity; i++)
      if (array[i].value == 0)
         print i
       else
         print  i ; array[i].key;array[i].value
      endif
endfor

Kamus/Deklarasi Variabel fungsi size_of_hashtable

Algoritma/Deskripsi fungsi size_of_hashtable
print size
```

```
Kamus/Deklarasi Variabel fungsi utama
choice,key =int

Algoritma/Deskripsi fungsi utama
struct data{key,value}
struct data *array
init_array()
   do
input choice;
      switch (choice)
         case 1:
            input key
            insert(key)
            break
         case 2:
            input key
            remove_element(key)
            break
         case 3:
            size_of_hashtable()
            break
         case 4:
            display()
            break
         default:
            print "Salah input."
       endswitch
      input choice
  enddo
    while (choice == 1)
free(array)
print 0
```

Algoritma :

1. Membuat fungsi hashcode(key)
2. Mencetak key%capacity
3. Membuat fungsi if_prime(n)
4. Jika(n<-1) maka kerjakan baris 5
5. Mencetak 0
6. Selama (i = 2) maka kerjakan baris 7 s.d 9
7. Jika(n%i==0)
8. Mencetak 0
9. i++
10. Mencetak 1
11. Membuat fungsi get_prime(n)
12. Jika(n%2==0) maka kerjakan baris 13
13. n++
14. Selama(!if_prime(n)) n+=2
15. Mencetak n
16. Membuat fungsi init_array
17. capacity = get_prime(capacity)
18. array = (struct data*)malloc(capacity * sizeof(struct data))
19. Selama (int i = 0) maka kerjakan baris 20 s.d 22
20. array[i].key = 0;
21. array[i].value = 0
22. i++
23. Membuat fungsi insert(key)
24. index = hashcode(key)
25. Jika (array[index].value==0) maka kerjakan baris 26 s.d 33
26. array[index].key=key
27. array[index].value=1
28. size++
29. Mencetak Nilai key
30. Jika(array[index].key == key) maka kerjakan baris 31 s.d 33
31. array[index].value += 1
32. Mencetak Nilai key
33. Mencetak “Elemen tidak dapat diinsert'
34. Membuat fungsi remove_element(key)
35. index = hashcode(key)
36. Jika (array[index].value==0) maka kerjakan baris 37 s.d 41
37. Mencetak “Kunci tidak ada”
38. array[index].key = 0
39. array[index].value = 0
40. size--
41. Mencetak Nilai key
42. Membuat fungsi display
43. Selama (i=0) maka kerjakan baris 44 s.d
44. Jika(array[i].value==0) maka kerjakan baris 45 s.d 45 kalau tidak kerjakan baris 46 s.d 48
45. Mencetak Nilai i
46. Mencetak Nilai i
47. Mencetak Nilai array[i].key
48. Mencetak Nilai array[i].value
49. Membuat fungsi size_of_hashtable
50. Mencetak size

51. Membuat fungsi utama
52. Memanggil fungsi init_array
53. Memasukkan Nilai choice
54. switch(choice)
55. case 1:
56. Memasukkan Nilai key;
57. Memanggil fungsi insert(key);
58. break
59. case 2:
60. Memasukkan Nilai key
61. remove_element(key)
62. break
63. case 3:
64. Memanggil fungsi size_of_hashtable()
65. break
66. case 4:
67. display()
68. break;
69. default:
70. Mencetak "Salah input."
71. Memasukkan Nilai choice
72. Selama (choice == 1)
73. free(array)
74. Mencetak 0
75. Selesai