



University of Asia Pacific

Department of Computer Science & Engineering

Compiler Design Lab

CSE 430

Submitted to:

Baivab Das

Lecturer

CSE, University of Asia Pacific

Submitted by:

Asma Sultana

20101084

Section: B1

Problem

Construct a simple hash-based symbol table (data dictionary) based on Chaining.

Description of the problem

We will build a symbol table in this assignment. At this initial stage, we will omit many details regarding an actual symbol table and simply add here to the basic concept that "a symbol table is an efficient data dictionary for the symbols used in a program". Thus, our assignment focuses on constructing a simple hash-based data dictionary based on chaining.

Description of the exercise code

I store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces, etc.

Here, I take the user input of a sequence of six tuples/attributes. They are

- Name
- Type
- Size
- Dimension
- Line of Code
- Address

for implementation add these global functions:

- (a) insert()
- (b) search()
- (c) delete()
- (d) show()
- (e) update()
- (f) getHashKey()

Code

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

using namespace std;

struct SymbolData
{
    string name;
    string type;
    string size;
    string dimension;
    string line_of_code;
    string line_of_usage;
    string address;
};

vector<SymbolData> symbol_table;

void insert(string name, string type, string size, string dimension, string line_of_code,
string line_of_usage, string address)
{
    SymbolData data = {name, type, size, dimension, line_of_code, line_of_usage,
address};
    symbol_table.push_back(data); //lambda function that serves as the comparison
function for sorting.
    sort(symbol_table.begin(), symbol_table.end(), [](const SymbolData& a, const
SymbolData& b)
    {
        return a.name < b.name;
    });
}

void search(string name)
{
```

```

        cout << "Index\tName\tType\tSize\tDimension\tLine of Code\tLine of
Usage\tAddress\n";
        for (size_t i = 0; i < symbol_table.size(); ++i)
        {
            if (symbol_table[i].name == name)
            {
                cout << i << "\t" << symbol_table[i].name << "\t" << symbol_table[i].type << "\t"
                    << symbol_table[i].size << "\t" << symbol_table[i].dimension << "\t\t"
                    << symbol_table[i].line_of_code << "\t\t" << symbol_table[i].line_of_usage <<
"\t\t"
                    << symbol_table[i].address << "\n";
                cout << "Data found!\n";
                return;
            }
        }
        cout << "Data not found for name: " << name << "\n";
    }

```

```

void update(string old_name, string new_name, string old_type, string new_type, string
old_size, string new_size,
            string old_dimension, string new_dimension, string old_line_of_code, string
new_line_of_code,
            string old_line_of_usage, string new_line_of_usage, string old_address, string
new_address)
{
    for (auto& data : symbol_table)
    {
        if (data.name == old_name)
        {
            data.name = new_name;
        }
        if (data.type == old_type)
        {
            data.type = new_type;
        }
        //
    }
}

```

```
void deleteData(string name)
```

```
{
    for (auto it = symbol_table.begin(); it != symbol_table.end(); ++it)
    {
        if (it->name == name)
        {
            symbol_table.erase(it);
            return;
        }
    }
    cout << "Name not found\n";
}
```

```
void show()
```

```
{
    cout << "Index\tName\tType\tSize\tDimension\tLine of Code\tLine of Usage\tAddress\n";
    for (size_t i = 0; i < symbol_table.size(); ++i)
    {
        cout << i << "\t" << symbol_table[i].name << "\t" << symbol_table[i].type << "\t"
            << symbol_table[i].size << "\t" << symbol_table[i].dimension << "\t\t"
            << symbol_table[i].line_of_code << "\t\t" << symbol_table[i].line_of_usage <<
            "\t\t"
            << symbol_table[i].address << "\n";
    }
}
```

```
int getHashKey(string name)
```

```
{
    for (size_t i = 0; i < symbol_table.size(); ++i)
    {
        if (symbol_table[i].name == name)
        {
            return i;
        }
    }
    return -1; // Not found
}
```

```

int main()
{
    while (true)
    {
        cout << "Press 0 to Exit the program\n";
        cout << "Press 1 to Insert data\n";
        cout << "Press 2 to Search any data\n";
        cout << "Press 3 to Update any data\n";
        cout << "Press 4 to Delete any data\n";
        cout << "Press 5 to Show all data\n";
        cout << "Press 6 to Get Hash Key\n\n";

        int choice;
        cout << "Press -> ";
        cin >> choice;
        cout << "\n";

        if (choice == 0)
        {
            cout << "Exiting the program. Thank you!\n";
            break;
        }

        if (choice == 1)
        {
            int n;
            std::cout << "How many inputs? - ";
            std::cin >> n;
            std::cout << "\n";

            for (int i = 0; i < n; ++i)
            {
                string name, type, address;
                string size, dimension, line_of_code, line_of_usage;
                cout << "Enter Name, Type, Size, Dimension, Line of Code, Line of Usage,
Address: ";
                cin >> name >> type >> size >> dimension >> line_of_code >> line_of_usage >>
address;

```

```

        insert(name, type, size, dimension, line_of_code, line_of_usage, address);
    }
    cout << "Data inserted successfully.\n\n";
}

if (choice == 2)
{
    string name;
    cout << "Enter Name: ";
    cin >> name;
    search(name);
    cout << "\n";
}

if (choice == 3)
{
    string old_name, new_name, old_type, new_type, old_address, new_address;
    string old_size, new_size, old_dimension, new_dimension, old_line_of_code,
new_line_of_code, old_line_of_usage, new_line_of_usage;

    cout << "Enter old name: ";
    cin >> old_name;
    cout << "Enter new name: ";
    cin >> new_name;

    cout << "Enter old type: ";
    cin >> old_type;
    cout << "Enter new type: ";
    cin >> new_type;

/*
    cout << "Enter old size: ";
    cin >> old_size;
    cout << "Enter new size: ";
    cin >> new_size;

    cout << "Enter old dimension: ";
    cin >> old_dimension;
    cout << "Enter new dimension: ";
    cin >> new_dimension;

```

```

        cout << "Enter old line of code: ";
        cin >> old_line_of_code;
        cout << "Enter new line of code: ";
        cin >> new_line_of_code;

        cout << "Enter old line of usage: ";
        cin >> old_line_of_usage;
        cout << "Enter new line of usage: ";
        cin >> new_line_of_usage;

        cout << "Enter old address: ";
        cin >> old_address;
        cout << "Enter new address: ";
        cin >> new_address;
    */
        update(old_name, new_name, old_type, new_type, old_size, new_size,
old_dimension, new_dimension,
        old_line_of_code, new_line_of_code, old_line_of_usage, new_line_of_usage,
old_address, new_address);

        cout << "Data updated successfully.\n\n";
    }

    if (choice == 4)
    {
        string name;
        cout << "Enter Name: ";
        cin >> name;
        deleteData(name);
        cout << "Data deleted successfully.\n\n";
    }

    if (choice == 5)
    {
        show();
        cout << "\n";
    }
}

```



```
if (choice == 6)
{
    string name;
    cout << "Enter Name: ";
    cin >> name;
    int hashKey = getHashKey(name);
    if (hashKey != -1)
    {
        cout << "Hash Key: " << hashKey << "\n";
    }
    else
    {
        cout << "Name not found\n";
    }
    cout << "\n";
}

if (choice < 0 || choice > 6)
{
    cout << "Invalid choice. Please try again. \n\n";
}

return 0;
}
```

Sample input

For data insert,

How many data inputs? -> 5

john char 4 1 5 12 0x6dfed4

age int 2 0 3 5 0x7ffdd8

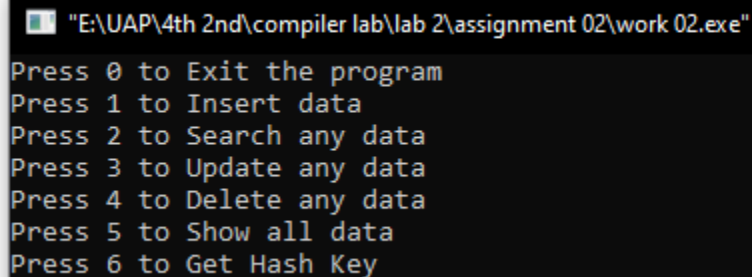
x ID 2 1 5 10 0x6dfed4

3 num 2 0 5 7 0x2dfg

gpa float 3 1 4 11 0x4aefdg

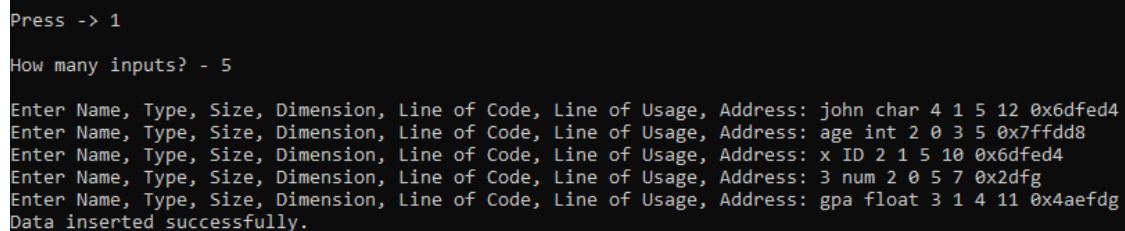
Observed output

When program code executes showing this template-



```
"E:\UAP\4th 2nd\compiler lab\lab 2\assignment 02\work 02.exe"
Press 0 to Exit the program
Press 1 to Insert data
Press 2 to Search any data
Press 3 to Update any data
Press 4 to Delete any data
Press 5 to Show all data
Press 6 to Get Hash Key
```

When insert operation executes-



```
Press -> 1
How many inputs? - 5
Enter Name, Type, Size, Dimension, Line of Code, Line of Usage, Address: john char 4 1 5 12 0x6dfed4
Enter Name, Type, Size, Dimension, Line of Code, Line of Usage, Address: age int 2 0 3 5 0x7ffdd8
Enter Name, Type, Size, Dimension, Line of Code, Line of Usage, Address: x ID 2 1 5 10 0x6dfed4
Enter Name, Type, Size, Dimension, Line of Code, Line of Usage, Address: 3 num 2 0 5 7 0x2dfg
Enter Name, Type, Size, Dimension, Line of Code, Line of Usage, Address: gpa float 3 1 4 11 0x4aefdg
Data inserted successfully.
```

When showing the data into symbol table-

```
Press -> 5
```

Index	Name	Type	Size	Dimension	Line of Code	Line of Usage	Address
0	3	num	2	0	5	7	0x2dfg
1	age	int	2	0	3	5	0x7ffdd8
2	gpa	float	3	1	4	11	0x4aefdg
3	john	char	4	1	5	12	0x6dfed4
4	x	ID	2	1	5	10	0x6dfed4

When search operation executes-

```
Press -> 2
```

Enter Name: age

Index	Name	Type	Size	Dimension	Line of Code	Line of Usage	Address
1	age	int	2	0	3	5	0x7ffdd8

Data found!

When update operation executes-

```
Press -> 3
```

Enter old name: x
Enter new name: y
Enter old type: ID
Enter new type: int
Data updated successfully.

Press 0 to Exit the program
Press 1 to Insert data
Press 2 to Search any data
Press 3 to Update any data
Press 4 to Delete any data
Press 5 to Show all data
Press 6 to Get Hash Key

```
Press -> 5
```

Index	Name	Type	Size	Dimension	Line of Code	Line of Usage	Address
0	3	num	2	0	5	7	0x2dfg
1	age	int	2	0	3	5	0x7ffdd8
2	gpa	float	3	1	4	11	0x4aefdg
3	john	char	4	1	5	12	0x6dfed4
4	y	int	2	1	5	10	0x6dfed4

When delete operation executes-

```
Press -> 4

Enter Name: age
Data deleted successfully.

Press 0 to Exit the program
Press 1 to Insert data
Press 2 to Search any data
Press 3 to Update any data
Press 4 to Delete any data
Press 5 to Show all data
Press 6 to Get Hash Key

Press -> 5
```

Index	Name	Type	Size	Dimension	Line of Code	Line of Usage	Address
0	3	num	2	0	5	7	0x2dfg
1	gpa	float	3	1	4	11	0x4aefdg
2	john	char	4	1	5	12	0x6dfed4
3	y	int	2	1	5	10	0x6dfed4

When hash key -

```
Press -> 6

Enter Name: john
Hash Key: 3
```

```
Press -> 6

Enter Name: 3
Hash Key: 0
```