Assignment 7

Title: To write a program for implementation of symbol table. And perform various operations.

Problem Statement: You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

Objectives:

- To understand the concept of symbol table.
- To understand need of symbol table.

Outcomes:

- We learnt use of symbol table.
- We learnt various methods of implementing symbol table.

Theory:

Symbol Table is an important data structure created and maintained by the compiler in order to keep track of semantics of variable i.e. it stores information about scope and binding information about names, information about instances of various entities such as variable and function names, classes, objects, etc.

- It is built in lexical and syntax analysis phases.
- The information is collected by the analysis phases of compiler and is used by synthesis phases of compiler to generate code.
- It is used by compiler to achieve compile time efficiency.
- It is used by various phases of compiler as follows:
    1. **Lexical Analysis**: Creates new table entries in the table, example like entries about token.
    2. **Syntax Analysis**: Adds information regarding attribute type, scope, dimension, line of reference, use, etc in the table.
    3. **Semantic Analysis**: Uses available information in the table to check for semantics i.e. to verify that expressions and assignments are semantically correct (type checking) and update it accordingly.

4. **Intermediate Code generation**: Refers symbol table for knowing how much and what type of run-time is allocated and table helps in adding temporary variable information.
5. **Code Optimization**: Uses information present in symbol table for machine dependent optimization.
6. **Target Code generation**: Generates code by using address information of identifier present in the table.

Class Definition:

```
class node
{
	char symbol[10];
	string attribute;
public:
	node(char* k)
	{
		strcpy(symbol, k);
		cout << "Attribute: ";
		getline(cin, attribute);
	}
	void show()
	{
		cout << symbol << "\t" << attribute << "\t";
	}
	friend class Hash;
};

class Hash
{
	node* arr[26];
	int chain[26];
public:
	Hash()
```

```
        {
                for (int i = 0; i < 26; i++)
                {
                        arr[i] = NULL;
                        chain[i] = -1;
                }
        }
        void create_wo_rep(int);
        void insert_wo_rep();
        void create_wrep(int);
        void insert_wrep();
        void disp();
        void del();
        void find();
        int ifpresent(int,char*);
        int full();
};
```

Pseudo Codes:

```
int Hash::full()
{
   int i;
   for(i=0;i<MAX && arr[i] != NULL;i++);
   if(i == MAX)
   {
      return 1;
   }
   return 0;
}

int Hash::ifpresent(int x,char* t)
{
```

```
        while(arr[x]->symbol[0] != t[0])
        {
                if(arr[x] == NULL)
                {
                        return 0;
                }
                x++;
        }
        return x;
}

void Hash::create_wo_rep(int n)
{
        char str[10];
        for(int i=0;i<n && !full();i++)
  {
     cout << "Symbol: ";
     cin.getline(str, 10);
     if(arr[(int) str[0] - 97] == NULL)
     {
        arr[(int) str[0] - 97] = new node(str);
     }
     else
     {
        char* temp = arr[(int)str[0]-97]->symbol;
        int j = (int) str[0] - 97;
        int o = (int) str[0] - 97;
        while(chain[j] != -1)
        {
           j = chain[j];
        }
```

```
            int p = j;
            while (arr[j] != NULL)
            {
               //j++;
               j = (j+1)%MAX;
            }
            arr[j] = new node(str);
            if(str[0] == temp[0])
            {
               chain[p] = j;
            }
            else
            {
               if(ifpresent(o,str) != o)
               {
                  p = ifpresent(o,str);
               }
               else
               {
                  p = j;
               }
            }
            chain[p] = j;
         }
      }
   }

void Hash::insert_wo_rep()
{
            char str[10];
            cout << "Symbol: ";
```

```
        cin.getline(str, 10);
        if(!full())
{
   if(arr[(int) str[0] - 97] == NULL)
   {
      arr[(int) str[0] - 97] = new node(str);
   }
   else
   {
      char* temp = arr[(int)str[0]-97]->symbol;
      int j = (int) str[0] - 97;
      int o = (int) str[0] - 97;
      while(chain[j] != -1)
      {
         j = chain[j];
      }
      int p = j;
      while (arr[j] != NULL)
      {
         //j++;
         j = (j+1)%MAX;
      }
      arr[j] = new node(str);
      if(str[0] == temp[0])
      {
         chain[p] = j;
      }
      else
      {
         if(ifpresent(o,str) != o)
         {
```

```
                p = ifpresent(o,str);
            }
            else
            {
                p = j;
            }
        }
        chain[p] = j;
    }
  }
}

void Hash::create_wrep(int n)
{
        char str[10];
        for(int i=0;i<n && !full();i++)
        {
            cout << "Symbol: ";
                cin.getline(str, 10);
    if(arr[(int)str[0]-97] == NULL)
    {
        arr[(int)str[0]-97] = new node(str);
    }
    else
    {
        char* temp = arr[(int)str[0]-97]->symbol;
        if(str[0] == temp[0])
        {
            int j = (int) str[0] - 97;
            while(j != -1 && chain[j] != -1)
            {
```

```
        j = chain[j];
    }
    int p = j;
    while (arr[j] != NULL)
    {
        j = (j+1)%MAX;
    }
    arr[j] = new node(str);
    chain[p] = j;
}
else
{
    int j = (int) str[0] - 97;
    int o = j;
    int l = (int) temp[0] - 97;
    //cout << "l: " << l;
    while(chain[l] != j)
    {
        l = chain[l];
    }
    //cout << "l: " << l;
    while(j != -1 && chain[j] != -1)
    {
        j = chain[j];
    }
    int p = j;
    while (arr[j] != NULL)
    {
        j++;
    }
    //cout << "j: " << j;
```

```
            arr[j] = arr[o];
            arr[o] = new node(str);
            chain[p] = j;
            chain[o] = -1;
            chain[l] = j;
          }
       }
           }
}

void Hash::insert_wrep()
{
           char str[10];
           cout << "Symbol: ";
           cin.getline(str, 10);
           if(!full())
   {
      if(arr[(int)str[0]-97] == NULL)
      {
         arr[(int)str[0]-97] = new node(str);
      }
      else
      {
         char* temp = arr[(int)str[0]-97]->symbol;
         if(str[0] == temp[0])
         {
            int j = (int) str[0] - 97;
            while(j != -1 && chain[j] != -1)
            {
               j = chain[j];
            }
```

```
    int p = j;
    while (arr[j] != NULL)
    {
        j = (j+1)%MAX;
    }
    arr[j] = new node(str);
    chain[p] = j;
}
else
{
    int j = (int) str[0] - 97;
    int o = j;
    int l = (int) temp[0] - 97;
    //cout << "l: " << l;
    while(chain[l] != j)
    {
        l = chain[l];
    }
    //cout << "l: " << l;
    while(j != -1 && chain[j] != -1)
    {
        j = chain[j];
    }
    int p = j;
    while (arr[j] != NULL)
    {
        j++;
    }
    //cout << "j: " << j;
    arr[j] = arr[o];
    arr[o] = new node(str);
```

```
            chain[p] = j;
            chain[o] = -1;
            chain[l] = j;
          }
        }
    }
}

void Hash::disp()
{
    cout << "Index\tSymbol\tAttribute\tChain\n";
          for (int i = 0; i < 26; i++)
          {
                if (arr[i] != NULL)
                {
                      cout << i << ".\t";
                      arr[i]->show();
                      cout << chain[i];
                      cout << "\n";
                }
          }
}

void Hash::find()
{
          cout << "Enter Symbol to be searched: ";
          char str[10];
          cin.getline(str,8);
          int j = (int) str[0] - 97;
          if(arr[j] == NULL)
          {
```

```
                cout << "Symbol not found.\n";
                return;
        }
        int i = -1,flag = 0;
        while(j != -1)
        {
                if(strcmp(arr[j]->symbol,str) == 0)
                {
                        flag = 1;
                        break;
                }
                i = j;
                j = chain[j];
        }
        if(flag == 0)
        {
                cout << "Symbol not found.\n";
                return;
        }
        if(i != -1)
        {
                chain[i] = chain[j];
        }
        arr[j]->show();
}

void Hash::del()
{
        cout << "Enter Symbol to be deleted: ";
        char str[10];
        cin.getline(str,8);
```

```
        int j = (int) str[0] - 97,i = -1,flag = 0;
        while(j != -1)
        {
                if(strcmp(arr[j]->symbol,str) == 0)
                {
                        flag = 1;
                        break;
                }
                i = j;
                j = chain[j];
        }
if(flag == 0)
        {
                cout << "Symbol not found.\n";
                return;
        }
if(i == -1)
{
   i = j;
   while(chain[j] != -1)
                {
                        j = chain[j];
                }
                arr[i] = arr[j];
                chain[i] = chain[j];
                arr[j] = NULL;
}
else
{
   chain[i] = chain[j];
   arr[j] = NULL;
```

```
    }
}
```

Test Cases:

1. Without Replacement.

2. With Replacement.

Enter choice: 2

1. Create Symbol Table.

2. Insert into Symbol Table.

3. Delete from Symbol Table.

4. Find from Symbol Table.

5. Display Symbol Table.

6. Exit.

Enter choice: 1

Enter number of Entries: 3

Symbol: abc

Attribute: abc

Symbol: aabc

Attribute: aabc

Symbol: bcd

Attribute: bcd

Enter choice: 5

Index	Symbol	Attribute	Chain
0.	abc	abc	2
1.	bcd	bcd	-1
2.	aabc	aabc	-1

Enter choice: 2

Symbol: cde

Attribute: cde

Enter choice: 5

Index Symbol Attribute Chain

0. abc abc 3

1. bcd bcd -1

2. cde cde -1

3. aabc aabc -1

Enter choice: 3

Enter Symbol to be deleted: abc

Enter choice: 5

Index Symbol Attribute Chain

0. aabc aabc -1

1. bcd bcd -1

2. cde cde -1

Enter choice: 4

Enter Symbol to be searched: bcd

bcd bcd

Enter choice: 6

Conclusion:

Symbol table was implemented successfully using C++.