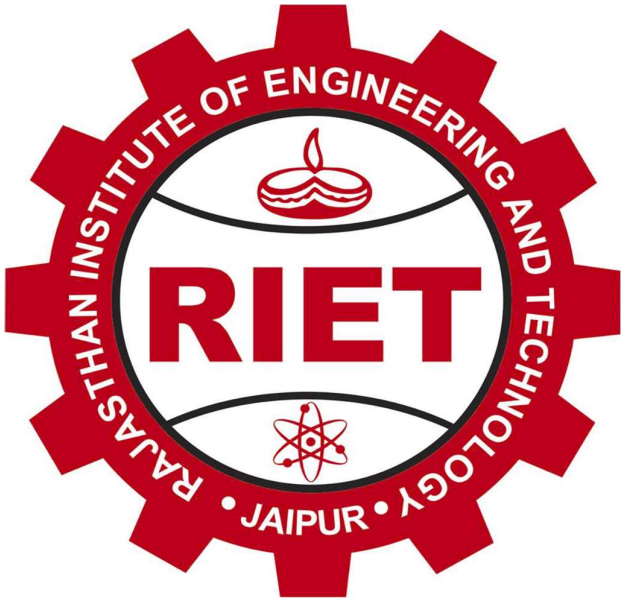
Rajasthan Institute Of Engineering and Technology



Lab- Compiler Design

Submitted to- Mr. Mukesh Chaudhary

Submitted by- Anshika Rai

Experiment - 6

AIM:

To write a program for implementing Symbol Table using C.

ALGORITHM:

Step1: Start the program for performing insert, display, delete, search and modify option in symbol table

Step2: Define the structure of the Symbol Table

Step3: Enter the choice for performing the operations in the symbol Table

Step4: If the entered choice is 1, search the symbol table for the symbol to be inserted. If the symbol is already present, it displays “Duplicate Symbol”. Else, insert the symbol and the corresponding address in the symbol table.

Step5: If the entered choice is 2, the symbols present in the symbol table are displayed.

Step6: If the entered choice is 3, the symbol to be deleted is searched in the symbol table.

Step7: If it is not found in the symbol table it displays “Label Not found”. Else, the symbol is deleted.

Step8: If the entered choice is 5, the symbol to be modified is searched in the symbol table.

PROGRAM CODE:

//Implementation of symbol table

```
      #include<stdio.h>
      #include<ctype.h>
      #include<stdlib.h>
      #include<string.h>
      #include<math.h>

void main()
{
      int i=0,j=0,x=0,n;
      void *p,*add[5];
      char ch,srch,b[15],d[15],c;
      printf("Expression terminated by $:");
      while((c=getchar())!='$')
      {
            b[i]=c;
            i++;
      }
      n=i-1;
      printf("Given Expression:");
      i=0;
      while(i<=n)
      {
            printf("%c",b[i]);
            i++;
      }
      printf("\n Symbol Table\n");
      printf("Symbol \t addr \t type");
      while(j<=n)
      {
            c=b[j];
            if(isalpha(toascii(c)))
            {
                  p=malloc(c);
                  add[x]=p;
                  d[x]=c;
                  printf("\n%c \t %d \t identifier\n",c,p);
                  x++;
                  j++;
            }
            Else
```

```
            {
                  ch=c;
                  if(ch=='+'||ch=='-'||ch=='*'||ch=='=')
                  {
                        p=malloc(ch);
                        add[x]=p;
                        d[x]=ch;
                        printf("\n %c \t %d \t operator\n",ch,p);
                        x++;
                        j++;
                  }
            }
      }
}
```

OUTPUT: