**Experiment No – 2**

**AIM: To implement a Symbol table with functions to create, insert, modify, search and displaying C language.**

ALGORITHM:

1. Start the program
2. Define the structure of the symbol table.
3. Enter the choice for performing the operations in the symbol table.
4. If choice is 1, search symbol table for the symbol to be inserted. If the symbol is already present display “Duplicate Symbol”, else insert symbol and corresponding address in the symbol table.
5. If choice is 2, symbols present in the symbols table are displayed.
6. If choice is 3, symbol to be deleted us searched in the symbol table. If found deletes else Displays “Not found”.
7. If choice is 5, the symbol to be modified is searched in the symbol table. The label or address or both can be modified.

PROGRAM:

/\* C program to implement SYMBOL TABLE \*/

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <string.h>

#define null 0

int size = 0;

void insert();

void del();

int mysearch(char[]);

void modify();

void display();

struct symboltab

{

char label[10];

int addr;

struct symboltab \*next;

};

struct symboltab \*first, \*last;

void main(){

int op;

int y;

char la[10];

do{

printf("\nSYMBOL TABLE IMPLEMENTATION\n");

printf("1. INSERT\n");

printf("2. Display\n");

printf("3. Delete\n");

printf("4. Search\n");

printf("5. Modify\n");

printf("6. End\n");

printf("Enter your options: ");

scanf("%d",&op);

switch (op){

case 1:

insert();

display();

break;

case 2:

display();

break;

case 3:

del();

display();

case 4:

printf("Enter the label to be searched: ");

scanf("%s",la);

y = mysearch(la);

if(y == 1)

printf("The label is already in the symbol table\n");

else

printf("The label is not found in the symbol table\n");

break;

case 5:

modify();

display();

case 6:

break;

}

}while(op<6);

getch();

}

void insert(){

int n;

char l[10];

printf("Enter the label: ");

scanf("%s",l);

n = mysearch(l);

if (n == 1)

printf("The label is already in the symbol table. Duplicate cannot be inserted\n");

else{

struct symboltab \*p;

p = malloc(sizeof(struct symboltab));

strcpy(p->label,l);

printf("Enter the address: ");

scanf("%d",&p->addr);

p->next = null;

if (size == 0){

first = p;

last = p;

}

else{

last->next = p;

last = p;

}

size++;

}

}

void display(){

int i;

struct symboltab \*p;

p = first;

printf("Label\t\tAddress\n");

for(i=0; i<size; i++){

printf("%s\t%d\n",p->label,p->addr);

p = p->next;

}

}

int mysearch(char lab[]){

int i;

int flag = 0;

struct symboltab \*p;

p = first;

for (i =0 ; i<size ; i++){

if(strcmp(p->label,lab) == 0){

flag = 1;

}

p = p->next;

}

return flag;

}

void modify(){

char l[10],nl[10];

int addr, choice, i, s;

struct symboltab \*p;

p = first;

printf("What do you want to modify?\n");

printf("1. Only the label\n");

printf("2. Only the address of a particular label\n");

printf("3. Both the label and address\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice){

case 1:

printf("Enter the old label\n");

scanf("%s",l);

printf("Enter the new label: ");

scanf("%s",nl);

s = mysearch(l);

if(s==0)

printf("\n No such label");

else{

for(i = 0; i<size ; i++){

if(strcmp(p->label,l)==0){

strcpy(p->label,nl);

}

p = p->next;

}

}

break;

case 2:

printf("Enter the label whose address is to be modified: ");

scanf("%s",l);

printf("Enter the new address: ");

scanf("%d",&addr);

s = mysearch(l);

if (s == 0)

printf("\nNo such Label");

else{

for(i = 0; i<size ; i++){

if(strcmp(p->label,l)==0){

p->addr = addr;

}

p = p->next;

}

}

break;

case 3:

printf("Enter the old label: ");

scanf("%s",l);

printf("Enter the new label: ");

scanf("%s",nl);

printf("Enter the new address: ");

scanf("%d",&addr);

s = mysearch(l);

if(s == 0)

printf("\nNo such label");

else{

for(i=0; i<size; i++){

if(strcmp(p->label,l)==0){

strcpy(p->label,nl);

p->addr = addr;

}

p = p->next;

}

}

break;

}

}

void del(){

int a;

char l[10];

struct symboltab \*p, \*q;

p = first;

printf("Enter the label to be deleted: ");

scanf("%s",l);

a = mysearch(l);

if (a == 0)

printf("Label not found! \n");

else{

if(strcmp(first->label,l)==0)

first = first->next;

else{

if(strcmp(last->label,l)==0)

{

q = p->next;

while(strcmp(q->label,l)!= 0){

p=p->next;

q = q->next;

}

p->next = null;

last = p;

}

else{

q= p->next;

while(strcmp(q->label,l)!=0)

{

p=p->next;

q=q->next;

}

p->next = q->next;

}

size--;

}

}

}

**Experiment no -3**

**IMPLEMENTATION OF SINGLE PASS ASSEMBLER**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <conio.h>

struct symtab

{

char symbol[20];

int addr;

char ch

}stab[20];

void main()

{

char cc[10],opcode[25],label[25],operand[25],object[10];

int s, entry = 0, flag = 0, c=0, i, op,locctr, code,length;

char mne[20][20] = {"LDA","STA","ADD","COMP","J","JLT"};

char opval[20][20] = {"1A","2B","1E","33","L9"};

FILE \*fp1, \*fp2;

fp1 = fopen("sinput.txt","r");

fp2 = fopen("sobj.txt","w");

fscanf(fp1,"%s%s",&label,&opcode);

if(strcmp(opcode,"START")==0)

fscanf(fp1,"%d",&locctr);

s = locctr;

fscanf(fp1,"%s%s%s",&label,&opcode,&operand);

while(!eof(fp1))

{

if(strcmp(opcode,"RESW")==0){

op = atoi(operand);

locctr = locctr + (op\*3);

}

else if(strcmp(opcode,"RESB")==0){

op = atoi(operand);

locctr = locctr+op;

}

else{

if(strcmp(opcode,"BYTE")==0){

op = strlen(operand);

locctr = locctr+op;

}

else if(strcmp(opcode,"WORD")==0){

locctr = locctr+3;

code = atoi(operand);

}

else

locctr = locctr+3;

fscanf(fp1,"%s%s%s",&label,&opcode,&operand);

}

length = locctr-s;

rewind(fp1);

fscanf(fp1,"%s%s",&label,&opcode);

printf("%s%s",label,opcode);

getch();

if(strcmp(opcode,"START") == 0)

fscanf(fp1,"%d",&locctr);

s = locctr;

fprintf(fp2,"H^%s^00%d^00%d",label,locctr,length);

fprintf(fp2,"\nT^%d^IE",locctr);

fscanf(fp1,"%s%s%s",&label,&opcode,&operand);

while(!feof(fp1))

{

strcpy(object,"");

flag = 0;

if(strcmp(label,"-")!=0){

entry = 0;

for(i=0;i<c;i++){

if(strcmp(stab[i].symbol,label)==0 && stab[i].ch == '\*')

{

fprintf(fp2,"\nT^%d^02^%d",stab[i].addr,locctr);

stab[c].addr = locctr;

stab[c].ch = 'd';

entry = 1;

fprintf(fp2,"\nT^%d^IE",locctr);

}

}

if(entry == 0 ){

strcpy(stab[c].symbol,label);

stab[c].addr = locctr;

stab[c].ch = 'd';

}

c++;

}

entry =0;

for(i =0 ;i<7;i++){

if(strcmp(opcode,mne[i])==0)

{

strcpy(object,opval[i]);

entry = 1;

break;

}

}

if(entry == 1)

{

for(i=0; i<c ; i++)

{

if(strcmp(operand,stab[i].symbol)==0){

code = stab[i].addr;

flag =1;

break;

}

}

if(flag == 0)

{

code = 0;

strcpy(stab[c].symbol,operand);

stab[c].addr = locctr+1;

stab[c].ch = '\*';

c++;

}

}

if(strcmp(opcode,"RESW")==0){

op=atoi(operand);

locctr = locctr+(op\*3);

}

else{

if(strcmp(opcode,"RESB")==0){

op = atoi(operand);

locctr = locctr+op;

}

else{

if(strcmp(opcode,"BYTE")==0){

op = strlen(operand);

locctr = locctr+op;

}

else if(strcmp(opcode,"WORD")==0){

locctr = locctr+3;

code = atoi(operand);

}

else{

locctr = locctr +3;

if(strcmp(opcode,"RESW")!=0 && strcmp(opcode,"RESB")!=0){

if(strcmp(opcode,"BYTE")==0){

fprintf(fp2,"^%s",object);

for(i=2;i<op-1;i++)

fprintf(fp2,"%d",toascii(operand[i]));

}

else{

fprintf(fp2,"^%s%d",object,code);

}

fscanf(fp1,"%s%s%s",&label,&opcode,&operand);

}

fprintf(fp2,"\nE^%d",s);

fclose(fp1);

fclose(fp2);

}

}

}

}

}

}

**Experiment no -4**

**Implementation of pass 2 compiler**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

void main()

{

char a[10],ad[10],label[10],opcode[10],operand[10],mnemonic[10],symbol[10];

int i,locctr,code,add,len,actual\_len;

FILE \*fp1,\*fp2,\*fp3,\*fp4;

clrscr();

fp1=fopen("twoout.dat","w");

fp2=fopen("symtab.dat","r");

fp3=fopen("out.dat","r");

fp4=fopen("optab.dat","r");

fscanf(fp3,"%s%s%s",label,opcode,operand);

if(strcmp(opcode,"START")==0)

{

fprintf(fp1,"\t%s\t%s\t%s\n",label,opcode,operand);

fscanf(fp3,"%d%s%s%s",&locctr,label,opcode,operand);

}

while(strcmp(opcode,"END")!=0)

{

if(strcmp(opcode,"BYTE")==0)

{

fprintf(fp1,"%d\t%s\t%s\t%s\t",locctr,label,opcode,operand);

len=strlen(operand);

actual\_len=len-3;

for(i=2;i<(actual\_len+2);i++)

{

itoa(operand[i],ad,16);

fprintf(fp1,"%s",ad);

}

fprintf(fp1,"\n");

}

else if(strcmp(opcode,"WORD")==0)

{

len=strlen(operand);

itoa(atoi(operand),a,10);

fprintf(fp1,"%d\t%s\t%s\t%s\t00000%s\n",locctr,label,opcode,operand,a);

}

else if((strcmp(opcode,"RESB")==0)||(strcmp(opcode,"RESW")==0))

{

fprintf(fp1,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);

}

else

{

rewind(fp4);

fscanf(fp4,"%s%d",mnemonic,&code);

while(strcmp(opcode,mnemonic)!=0)

fscanf(fp4,"%s%d",mnemonic,&code);

if(strcmp(operand,"\*\*")==0)

{

fprintf(fp1,"%d\t%s\t%s\t%s\t%d0000\n",locctr,label,opcode,operand,code);

}

else

{

rewind(fp2);

fscanf(fp2,"%s%d",symbol,&add);

while(strcmp(operand,symbol)!=0)

{

fscanf(fp2,"%s%d",symbol,&add);

}

fprintf(fp1,"%d\t%s\t%s\t%s\t%d%d\n",locctr,label,opcode,operand,code,add);

}

}

fscanf(fp3,"%d%s%s%s\n",&locctr,label,opcode,operand);

}

fprintf(fp1,"%d\t%s\t%s\t%s\n",locctr,label,opcode,operand);

printf("FINISHED");

fclose(fp1);

fclose(fp2);

fclose(fp3);

fclose(fp4);

getch();

}