**CSA1443- Compiler Design for Intraprocedural Analysis**

**192321047 – KAUSHIK NARAYANAN.V**

**11)Implement a C program to perform symbol table operations.**

**Aim:**

To implement a symbol table in C that supports insertion, search, and display operations.

**Code:**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define SIZE 100

struct Symbol {

char name[50];

char type[20];

int address;

} table[SIZE];

int count = 0;

void insert(char name[], char type[], int address) {

strcpy(table[count].name, name);

strcpy(table[count].type, type);

table[count].address = address;

count++;

}

int search(char name[]) {

for (int i = 0; i < count; i++) {

if (strcmp(table[i].name, name) == 0)

return i;

}

return -1;

}

void display() {

printf("\nSymbol Table:\n");

printf("Name\tType\tAddress\n");

for (int i = 0; i < count; i++) {

printf("%s\t%s\t%d\n", table[i].name, table[i].type, table[i].address);

}

}

int main() {

insert("x", "int", 100);

insert("y", "float", 104);

insert("func", "function", 200);

display();

char searchName[50];

printf("\nEnter symbol to search: ");

scanf("%s", searchName);

int pos = search(searchName);

if (pos != -1)

printf("%s found at index %d\n", searchName, pos);

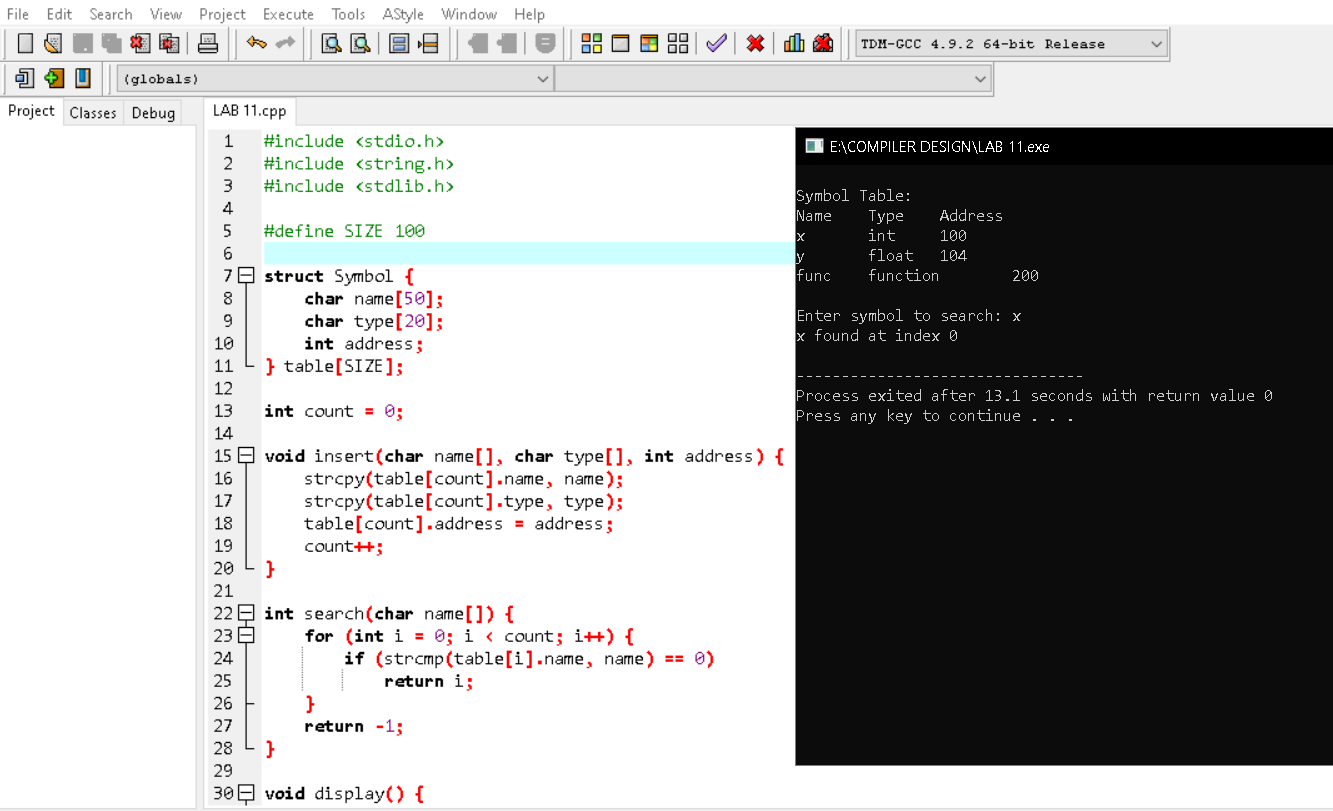
else

printf("%s not found in symbol table.\n", searchName);

return 0;

}

**Output:**

****

**12)Write a C program to construct recursive descent parsing for the given grammar**

**Aim:**

To implement a recursive descent parser for the given grammar.

**Code:**

#include <stdio.h>

#include <string.h>

char input[100];

int index = 0;

void E(), T(), F(), E\_prime(), T\_prime();

void match(char expected) {

if (input[index] == expected) {

index++;

} else {

printf("Error in parsing\n");

exit(0);

}

}

void E() {

T();

E\_prime();

}

void E\_prime() {

if (input[index] == '+') {

match('+');

T();

E\_prime();

}

}

void T() {

F();

T\_prime();

}

void T\_prime() {

if (input[index] == '\*') {

match('\*');

F();

T\_prime();

}

}

void F() {

if (input[index] == '(') {

match('(');

E();

match(')');

} else if (input[index] == 'i') {

match('i'); // Assuming 'i' represents an identifier

} else {

printf("Error in parsing\n");

exit(0);

}

}

int main() {

printf("Enter input string: ");

scanf("%s", input);

E();

if (input[index] == '\0') {

printf("Parsing successful\n");

} else {

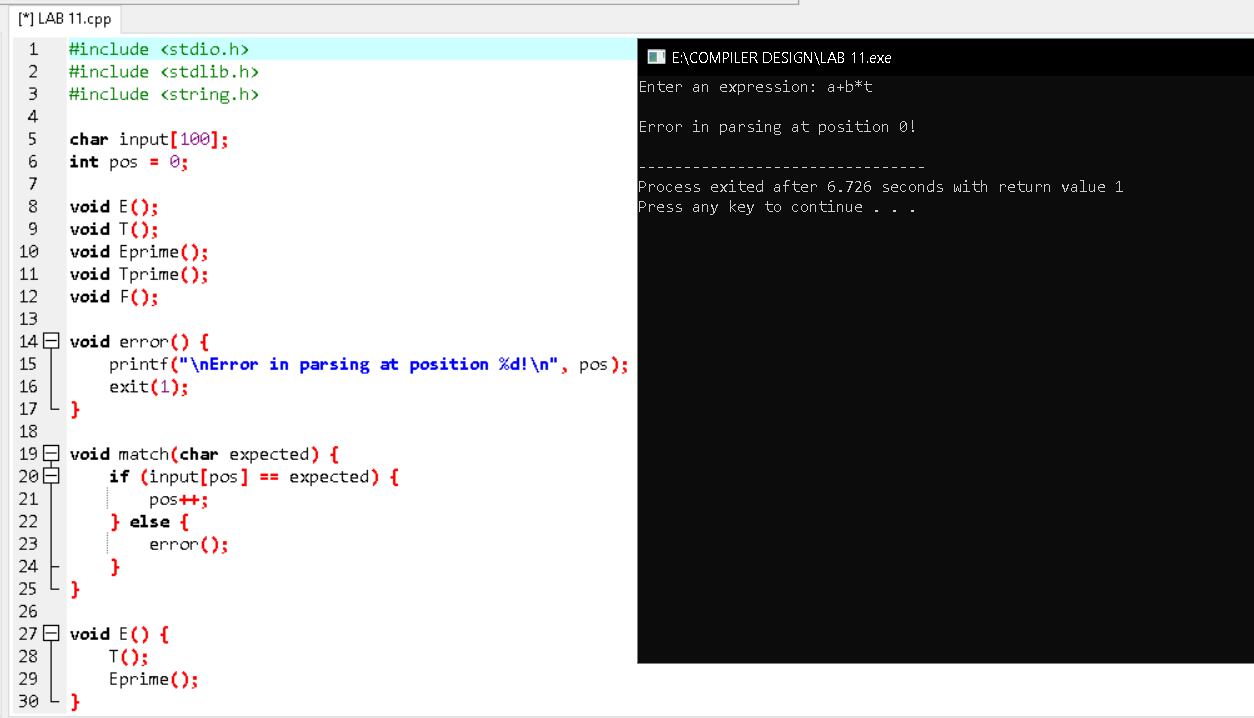
printf("Error: Unexpected input\n");

}

return 0;

}

**Output:**

****