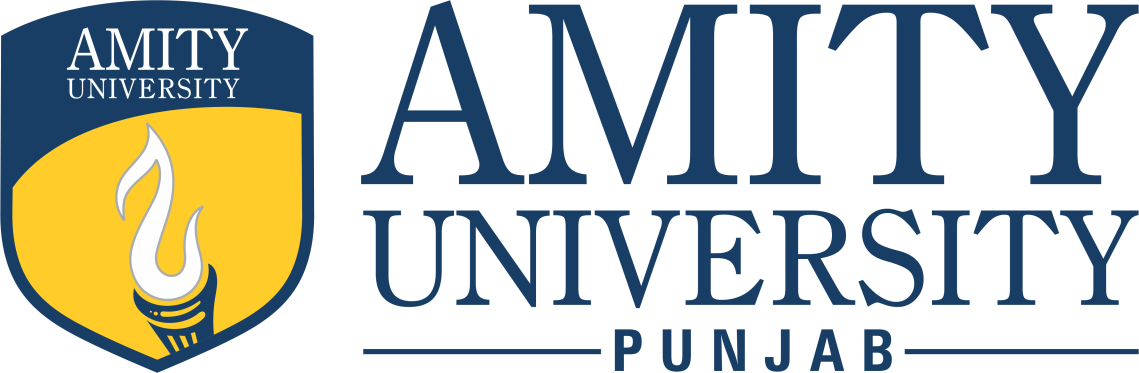
**PRACTICAL FILE**

**COMPILER DESIGNING**



**B. TECH-CSE-B**

**SEMESTER-6th**

**SUBMITTED TO: SUBMITTED BY:**

**DR. LAW KUMAR Jasmeet Kaur**

**ASET, AMITY UNIVERSITY BTech cse-6th SEM**

**MOHALI A25305222128**

1. Write a program in c for calculator

#include <stdio.h>

int main() {

char operator;

double num1, num2, result;

printf("Enter an operator (+, -, \*, /): ");

scanf(" %c", &operator);

printf("Enter two numbers: ");

scanf("%lf %lf", &num1, &num2);

switch(operator) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

if (num2 != 0)

result = num1 / num2;

else {

printf("Error! Division by zero is not allowed.\n");

return 1;

}

break;

default:

printf("Invalid operator!\n");

return 1;

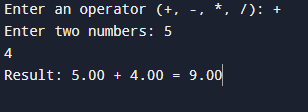
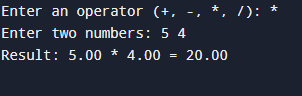
}

printf("Result: %.2lf %c %.2lf = %.2lf\n", num1, operator, num2, result);

return 0;

}

OUTPUT: -

1. Write a code in c for matrix addition code.

#include <stdio.h>

int main() {

int rows, cols;

printf("Enter the number of rows and columns: ");

scanf("%d %d", &rows, &cols);

int matrix1[rows][cols], matrix2[rows][cols], sum[rows][cols];

printf("Enter elements of first matrix:\n");

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

for (int j = 0; j < cols; j++) {

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements of second matrix:\n");

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

for (int j = 0; j < cols; j++) {

scanf("%d", &matrix2[i][j]);

}

}

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

for (int j = 0; j < cols; j++) {

sum[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

printf("Sum of matrices:\n");

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

for (int j = 0; j < cols; j++) {

printf("%d ", sum[i][j]);

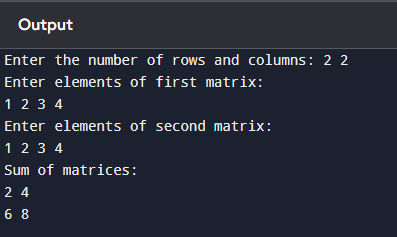
}

printf("\n");

}

return 0;

}



1. Write a code in c for matrix multiplication.

#include <stdio.h>

int main() {

int r1, c1, r2, c2;

printf("Enter number of rows and columns for first matrix: ");

scanf("%d %d", &r1, &c1);

printf("Enter number of rows and columns for second matrix: ");

scanf("%d %d", &r2, &c2);

printf("First matrix: %d x %d\n", r1, c1);

printf("Second matrix: %d x %d\n", r2, c2);

if (c1 != r2) {

printf("Matrix multiplication not possible. Columns of first matrix must match rows of second matrix.\n");

return 1;

}

int matrix1[r1][c1], matrix2[r2][c2], result[r1][c2];

printf("Enter elements of first matrix:\n");

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++) {

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements of second matrix:\n");

for (int i = 0; i < r2; i++) {

for (int j = 0; j < c2; j++) {

scanf("%d", &matrix2[i][j]);

}

}

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

result[i][j] = 0;

}

}

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

for (int k = 0; k < c1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

printf("Product of matrices (%d x %d):\n", r1, c2);

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

printf("%d ", result[i][j]);

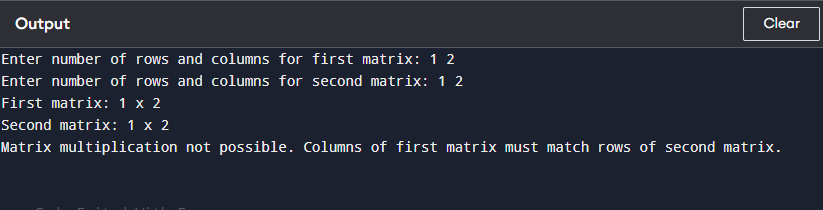
}

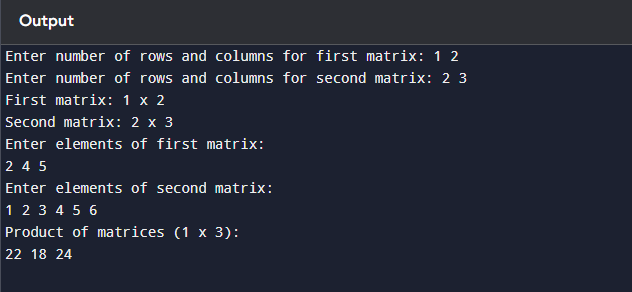
printf("\n");

}

return 0;

}





1. Write a code in c to check string is constant or not.

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int main() {

char inp[100];

printf("Enter the string to check whether it is a constant or not: ");

scanf("%s", inp);

int length = strlen(inp);

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

if (!isdigit(inp[i])) {

printf("Invalid! NOT A CONSTANT\n");

return 0;

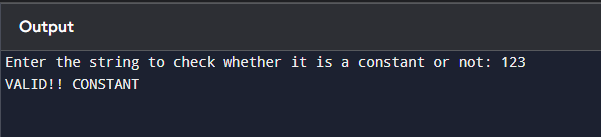
}

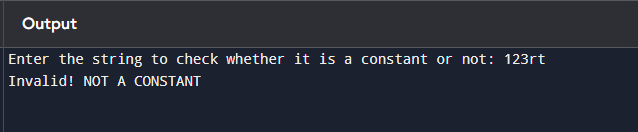
}

printf("VALID!! CONSTANT\n");

return 0;

}



****

1. Write a code in c to count no of lines and spaces.

#include <stdio.h>

int main() {

char ch;

int lines = 0, spaces = 0;

int enter\_count = 0;

printf("Enter text (press Enter twice to end input):\n");

while (1) {

ch = getchar();

if (ch == '\n') {

enter\_count++;

lines++;

if (enter\_count == 2) {

break;

}

} else {

enter\_count = 0;

if (ch == ' ') {

spaces++;

}

}

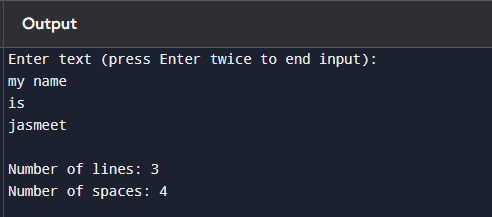
}

printf("Number of lines: %d\n", lines - 1);

printf("Number of spaces: %d\n", spaces);

return 0;

}



1. Write a code in c to check identifiers in c program.

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int isIdentifier(char \*str) {

if (!isalpha(str[0]) && str[0] != '\_')

return 0;

for (int i = 1; str[i] != '\0'; i++) {

if (!isalnum(str[i]) && str[i] != '\_')

return 0;

}

return 1;

}

int main() {

char input[100];

printf("Enter a string to check if it's a valid identifier: ");

scanf("%s", input);

if (isIdentifier(input))

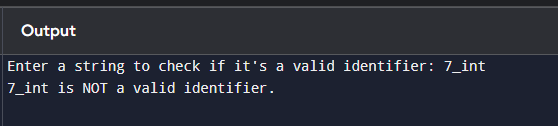
printf("%s is a valid identifier.\n", input);

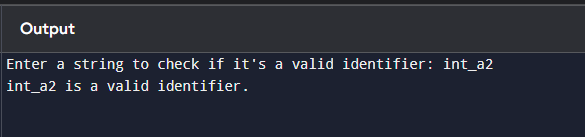
else

printf("%s is NOT a valid identifier.\n", input);

return 0;

}





1. Write a program to check the keywords in c program.

#include <stdio.h>

#include <string.h>

const char \*keywords[] = {"auto", "break", "case", "char", "const", "continue", "default", "do", "double", "else", "enum", "extern", "float", "for", "goto", "if", "inline", "int", "long", "register", "restrict", "return", "short", "signed", "sizeof", "static", "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"};

const int num\_keywords = sizeof(keywords) / sizeof(keywords[0]);

int isKeyword(char \*word) {

for (int i = 0; i < num\_keywords; i++) {

if (strcmp(word, keywords[i]) == 0) {

return 1;

}

}

return 0;

}

int main() {

char input[100];

printf("Enter a word to check if it's a C keyword: ");

scanf("%s", input);

if (isKeyword(input)) {

printf("%s is a C keyword.\n", input);

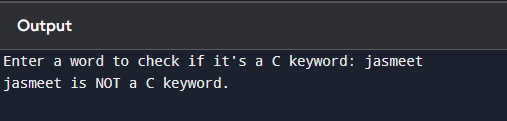
} else {

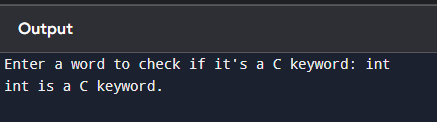
printf("%s is NOT a C keyword.\n", input);

}

return 0;

}





1. Write a menu-based program to check keywords, identifiers, space and constant in c (lexical analyser).

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_LEN 100

const char \*keywords[] = {"auto", "break", "case", "char", "const", "continue", "default", "do", "double", "else", "enum", "extern", "float", "for", "goto", "if", "inline", "int", "long", "register", "restrict", "return", "short", "signed", "sizeof", "static", "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"};

const int num\_keywords = sizeof(keywords) / sizeof(keywords[0]);

int isKeyword(char \*str) {

for (int i = 0; i < num\_keywords; i++) {

if (strcmp(str, keywords[i]) == 0) {

return 1;

}

}

return 0;

}

int isIdentifier(char \*str) {

if (!isalpha(str[0]) && str[0] != '\_')

return 0;

for (int i = 1; str[i] != '\0'; i++) {

if (!isalnum(str[i]) && str[i] != '\_')

return 0;

}

return 1;

}

int isConstant(char \*str) {

int i = 0;

if (str[i] == '-' || str[i] == '+') i++; // Allow signed numbers

for (; str[i] != '\0'; i++) {

if (!isdigit(str[i]))

return 0;

}

return i > 0;

}

void checkLexicalElements(char \*input) {

char temp[MAX\_LEN];

int j = 0;

for (int i = 0; input[i] != '\0'; i++) {

if (isspace(input[i]) || ispunct(input[i])) {

if (j > 0) {

temp[j] = '\0';

if (isKeyword(temp)) {

printf("%s is a Keyword\n", temp);

} else if (isIdentifier(temp)) {

printf("%s is an Identifier\n", temp);

} else if (isConstant(temp)) {

printf("%s is a Constant\n", temp);

} else {

printf("%s is an Unknown token\n", temp);

}

j = 0;

}

if (ispunct(input[i]) && input[i] != ' ') {

printf("%c is a Special Symbol\n", input[i]);

}

} else {

temp[j++] = input[i];

}

}

if (j > 0) {

temp[j] = '\0';

if (isKeyword(temp)) {

printf("%s is a Keyword\n", temp);

} else if (isIdentifier(temp)) {

printf("%s is an Identifier\n", temp);

} else if (isConstant(temp)) {

printf("%s is a Constant\n", temp);

} else {

printf("%s is an Unknown token\n", temp);

}

}

}

int main() {

char input[MAX\_LEN];

printf("Enter the input string: ");

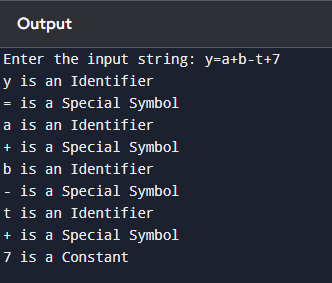
fgets(input, MAX\_LEN, stdin);

input[strcspn(input, "\n")] = '\0';

checkLexicalElements(input);

return 0;

}



1. Write a program for left recursion in c.

#include <stdio.h>

#include <string.h>

#define MAX 100

void eliminateLeftRecursion(char nonTerminal, char alpha[], char beta[]) {

char newNonTerminal = nonTerminal + ('\''); // Creating A'

printf("\nGiven Grammar:\n");

printf(" %c -> %c%c | %s\n", nonTerminal, nonTerminal, alpha[0], beta);

printf("\nGrammar after eliminating Left Recursion:\n");

printf(" %c -> %s%c\n", nonTerminal, beta, newNonTerminal);

printf(" %c' -> %s%c' | ε\n", nonTerminal, alpha, newNonTerminal);

}

int main() {

char nonTerminal;

char alpha[MAX], beta[MAX];

printf("Enter the non-terminal: ");

scanf(" %c", &nonTerminal);

printf("Enter the recursive part (α): ");

scanf("%s", alpha);

printf("Enter the non-recursive part (β): ");

scanf("%s", beta);

eliminateLeftRecursion(nonTerminal, alpha, beta);

return 0;

}

