Compiler Design(18CSC304J)

Experiment 12(a)

Quadruples, Triples, Indirect Triples

Harsh Goel
RA1811003010185

Aim: To study and implement Quadruples, Triples, Indirect Triples.

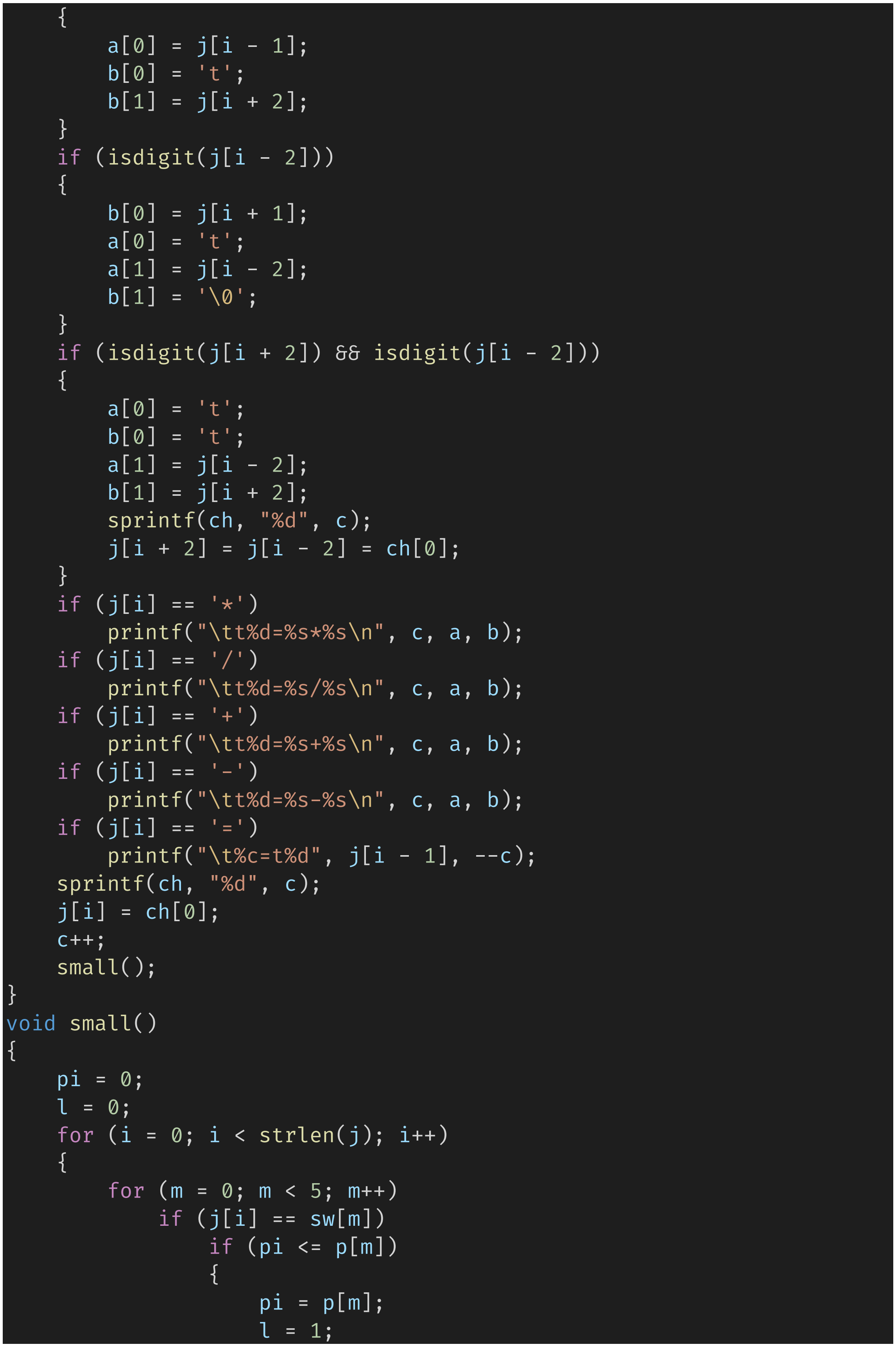
Language: C++

Problem:

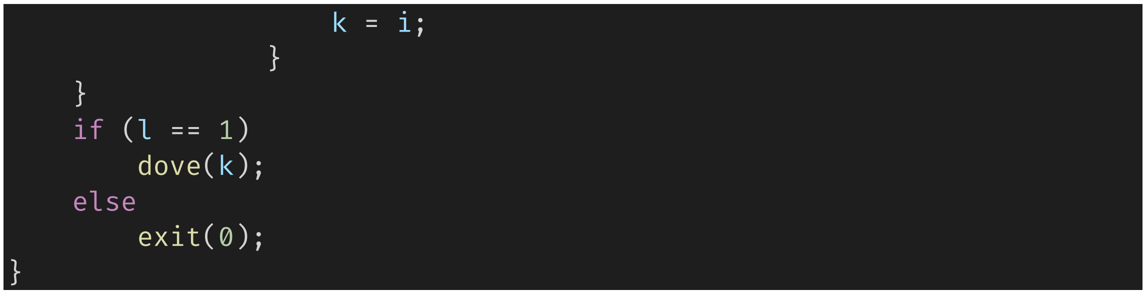
Implement the front end of a compiler that generates the three address code for a simple language with: one data type integer, arithmetic operators, relational operators, variable declaration statement, one conditional construct, one iterative construct and assignment statement.

Code Snippet:

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
void small();
void dove(int i);
int p[5] = {0, 1, 2, 3, 4}, c = 1, i, k, l, m, pi;
char sw[5] = {'=', '-', '+', '/', '*'}, j[20], a[5], b[5], ch[2];
void main()
{
    printf("Enter the expression:");
    scanf("%s", j);
    printf("\tThe Intermediate code is:\n");
    small();
}
void dove(int i)
{
    a[0] = b[0] = '\0';
    if (!isdigit(j[i + 2]) && !isdigit(j[i - 2]))
    {
        a[0] = j[i - 1];
        b[0] = j[i + 1];
    }
    if (isdigit(j[i + 2]))
```

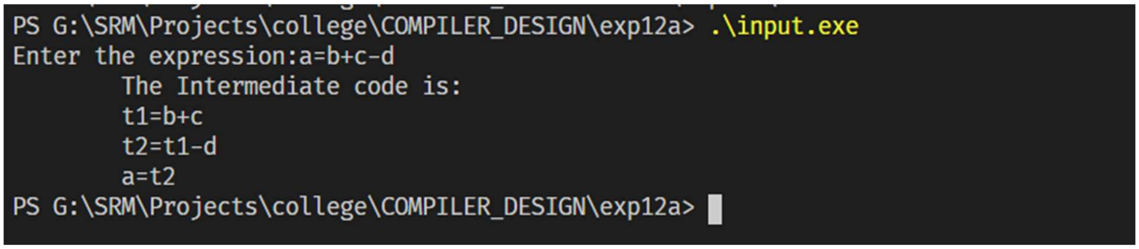


```
    {
        a[0] = j[i - 1];
        b[0] = 't';
        b[1] = j[i + 2];
    }
    if (isdigit(j[i - 2]))
    {
        b[0] = j[i + 1];
        a[0] = 't';
        a[1] = j[i - 2];
        b[1] = '\0';
    }
    if (isdigit(j[i + 2]) && isdigit(j[i - 2]))
    {
        a[0] = 't';
        b[0] = 't';
        a[1] = j[i - 2];
        b[1] = j[i + 2];
        sprintf(ch, "%d", c);
        j[i + 2] = j[i - 2] = ch[0];
    }
    if (j[i] == '*')
        printf("\tt%d=%s*%s\n", c, a, b);
    if (j[i] == '/')
        printf("\tt%d=%s/%s\n", c, a, b);
    if (j[i] == '+')
        printf("\tt%d=%s+%s\n", c, a, b);
    if (j[i] == '-')
        printf("\tt%d=%s-%s\n", c, a, b);
    if (j[i] == '=')
        printf("\t%c=t%d", j[i - 1], --c);
    sprintf(ch, "%d", c);
    j[i] = ch[0];
    c++;
    small();
}
void small()
{
    pi = 0;
    l = 0;
    for (i = 0; i < strlen(j); i++)
    {
        for (m = 0; m < 5; m++)
            if (j[i] == sw[m])
                if (pi <= p[m])
                {
                    pi = p[m];
                    l = 1;
```



```
                    k = i;
                }
    }
    if (l == 1)
        dove(k);
    else
        exit(0);
}
```

Output Screenshots:



```
PS G:\SRM\Projects\college\COMPILER_DESIGN\exp12a> .\input.exe
Enter the expression:a=b+c-d
        The Intermediate code is:
        t1=b+c
        t2=t1-d
        a=t2
PS G:\SRM\Projects\college\COMPILER_DESIGN\exp12a>
```

Result:

The code was successfully implemented and output was recorded.