

**Ex No: 9**

**Date:**

### **IMPLEMENT CODE OPTIMIZATION TECHNIQUES CONSTANT FOLDING**

**AIM:**

To write a C program to implement Constant Folding (Code optimization Technique).

**ALGORITHM:**

- The desired header files are declared.
- The two file pointers are initialized one for reading the C program from the file and one for writing the converted program with constant folding.
- The file is read and checked if there are any digits or operands present.
- If there is, then the evaluations are to be computed in switch case and stored.
- Copy the stored data to another file.
- Print the copied data file.

**PROGRAM:**

```
#include<stdio.h>
#include<string.h>
void main() {
    char s[20];
    char flag[20]="//Constant";
    char result,equal,operator;
    double op1,op2,interrslt;
    int a,flag2=0;
    FILE *fp1,*fp2;
    fp1 = fopen("input.txt","r");
    fp2 = fopen("output.txt","w");
    fscanf(fp1,"%s",s);
    while(!feof(fp1)) {
        if(strcmp(s,flag)==0) {
            flag2 = 1;
        }
        if(flag2==1) {
            fscanf(fp1,"%s",s);
            result=s[0];
            equal=s[1];
            if(isdigit(s[2])&& isdigit(s[4])) {
                if(s[3]=='+'||s[3]=='-'||s[3]=='*'||s[3]=='/') {
                    operator=s[3];
                    switch(operator) {
                        case '+':
                            interrslt=(s[2]-48)+(s[4]-48);
                            break;
                        case '-':
                            interrslt=(s[2]-48)-(s[4]-48);
```

```

        break;
    case '*':
        interrslt=(s[2]-48)*(s[4]-48);
        break;
    case '/':
        interrslt=(s[2]-48)/(s[4]-48);
        break;
    default:
        interrslt = 0;
        break; }
    fprintf(fp2,"/*Constant Folding*\n");
    fprintf(fp2,"%c = %lf\n",result,interrslt);
    flag2 = 0;
}
} else {
    fprintf(fp2,"Not Optimized\n");
    fprintf(fp2,"%s\n",s);
}
} else {
    fprintf(fp2,"%s\n",s);
}
}
fscanf(fp1,"%s",s);
}
fclose(fp1);
fclose(fp2);
}

```

### OUTPUT:

```

[VaruneshB210701303@localhost ~]$ vi input303.txt
[VaruneshB210701303@localhost ~]$ vi exp303.c
[VaruneshB210701303@localhost ~]$ cc exp303.c
[VaruneshB210701303@localhost ~]$ ./a.out
[VaruneshB210701303@localhost ~]$ vi output303.txt

```

//output303.txt

```

a=7
b=10
c=5
d=7

```

### RESULT:

**Ex No: 10**

**Date:**

## **IMPLEMENT CODE OPTIMIZATION TECHNIQUES DEAD CODE AND COMMON SUB EXPRESSION ELIMINATION**

**AIM:**

To write a C program to implement the dead code elimination and common sub expression elimination (code optimization) techniques.

**ALGORITHM:**

- Start
- Create the input file which contains three address code.
- Open the file in read mode.
- If the file pointer returns NULL, exit the program else go to 5.
- Scan the input symbol from left to right.
- Store the first expression in a string.
- Compare the string with the other expressions in the file.
- If there is a match, remove the expression from the input file.
- Perform these steps 5-8 for all the input symbols in the file.
- Scan the input symbol from the file from left to right.
- Get the operand before the operator from the three address code.
- Check whether the operand is used in any other expression in the three address code.
- If the operand is not used, then eliminate the complete expression from the three-address code else go to 14.
- Perform steps 11 to 13 for all the operands in the three address code till end of the file is reached.
- Stop.

**PROGRAM:**

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

struct op
{
    char l;
    char r[20];
}
op[10], pr[10];

void main()
{
    int a, i, k, j, n, z = 0, m, q;
```