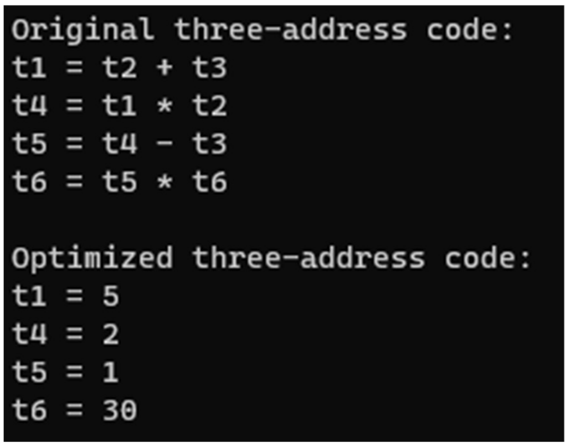
Program :-

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>

#define MAX_CODE_LINES 100
#define MAX_CODE_LENGTH 50

// Structure to represent a three-address code line
typedef struct {
    char op;  // Operation: '+', '-', '*', '/'
    int result;  // Result variable index
    int arg1;    // Argument 1 variable index or constant value
    int arg2;    // Argument 2 variable index or constant value
} ThreeAddressCode;

ThreeAddressCode code[MAX_CODE_LINES]; // Array to store three-address code lines
int codeCount = 0; // Current count of code lines

// Function to add a three-address code line to the array
void addCodeLine(char op, int result, int arg1, int arg2) {
    code[codeCount].op = op;
    code[codeCount].result = result;
    code[codeCount].arg1 = arg1;
    code[codeCount].arg2 = arg2;
    codeCount++;
}

// Function to perform basic optimizations on three-address code
void optimizeCode() {
    for (int i = 0; i < codeCount; i++) {
        // Check if both arguments are constants
        if (code[i].arg1 >= 0 && code[i].arg2 >= 0) {
            // Perform constant folding
            int value;
            switch (code[i].op) {
                case '+':
                    value = code[i].arg1 + code[i].arg2;
                    break;
                case '-':
                    value = code[i].arg1 - code[i].arg2;
                    break;
                case '*':
                    value = code[i].arg1 * code[i].arg2;
                    break;
                case '/':
                    value = code[i].arg1 / code[i].arg2;
                    break;
            }
            // Replace the current line with a simplified assignment if applicable
            code[i].op = '=';
            code[i].arg1 = value;
            code[i].arg2 = -1; // No need for a second argument in an assignment
        }
    }
}

// Function to print the optimized three-address code
void printOptimizedCode() {
    for (int i = 0; i < codeCount; i++) {
        if (code[i].op == '=') {
            printf("t%d = %d\n", code[i].result, code[i].arg1);
        } else {
            printf("t%d = t%d %c t%d\n", code[i].result, code[i].arg1, code[i].op, code[i].arg2);
        }
    }
}

int main() {
    // Example three-address code
    addCodeLine('+', 1, 2, 3);
    addCodeLine('*', 4, 1, 2);
    addCodeLine('-', 5, 4, 3);
    addCodeLine('*', 6, 5, 6);

    printf("Original three-address code:\n");
    printOptimizedCode();

    printf("\nOptimized three-address code:\n");
    optimizeCode();
    printOptimizedCode();

    return 0;
}
```

Output :-



```
Original three-address code:
t1 = t2 + t3
t4 = t1 * t2
t5 = t4 - t3
t6 = t5 * t6

Optimized three-address code:
t1 = 5
t4 = 2
t5 = 1
t6 = 30
```