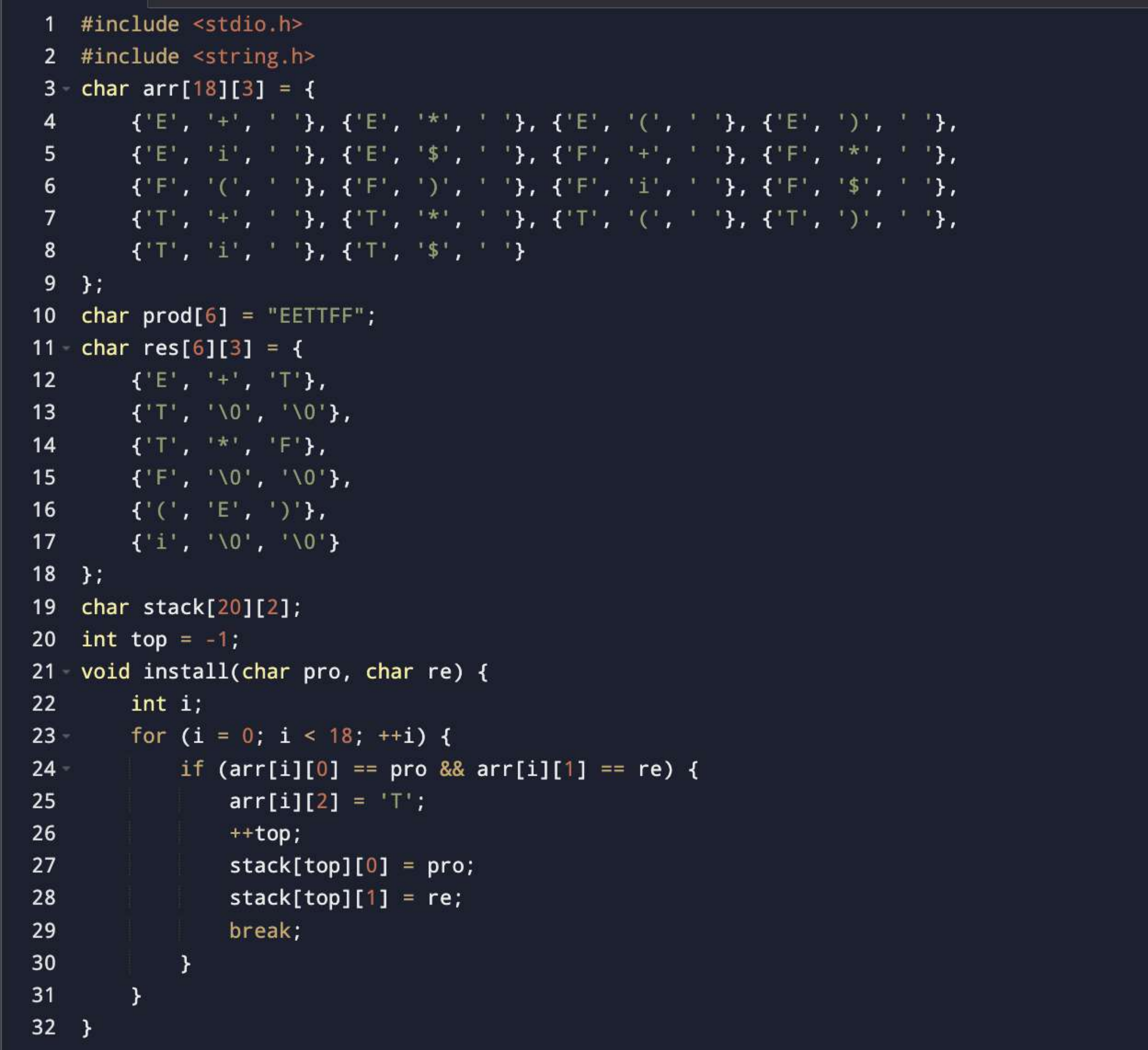
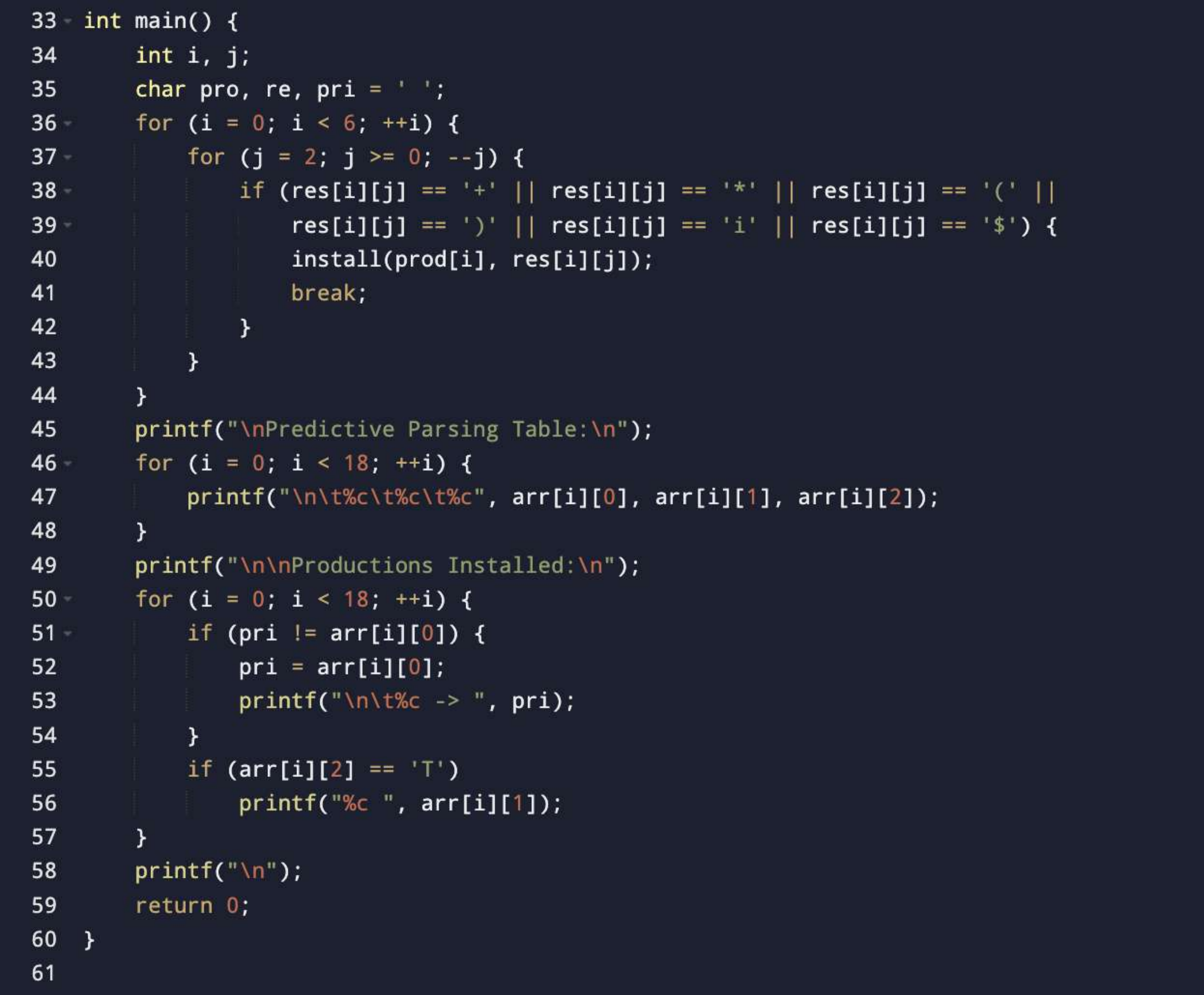
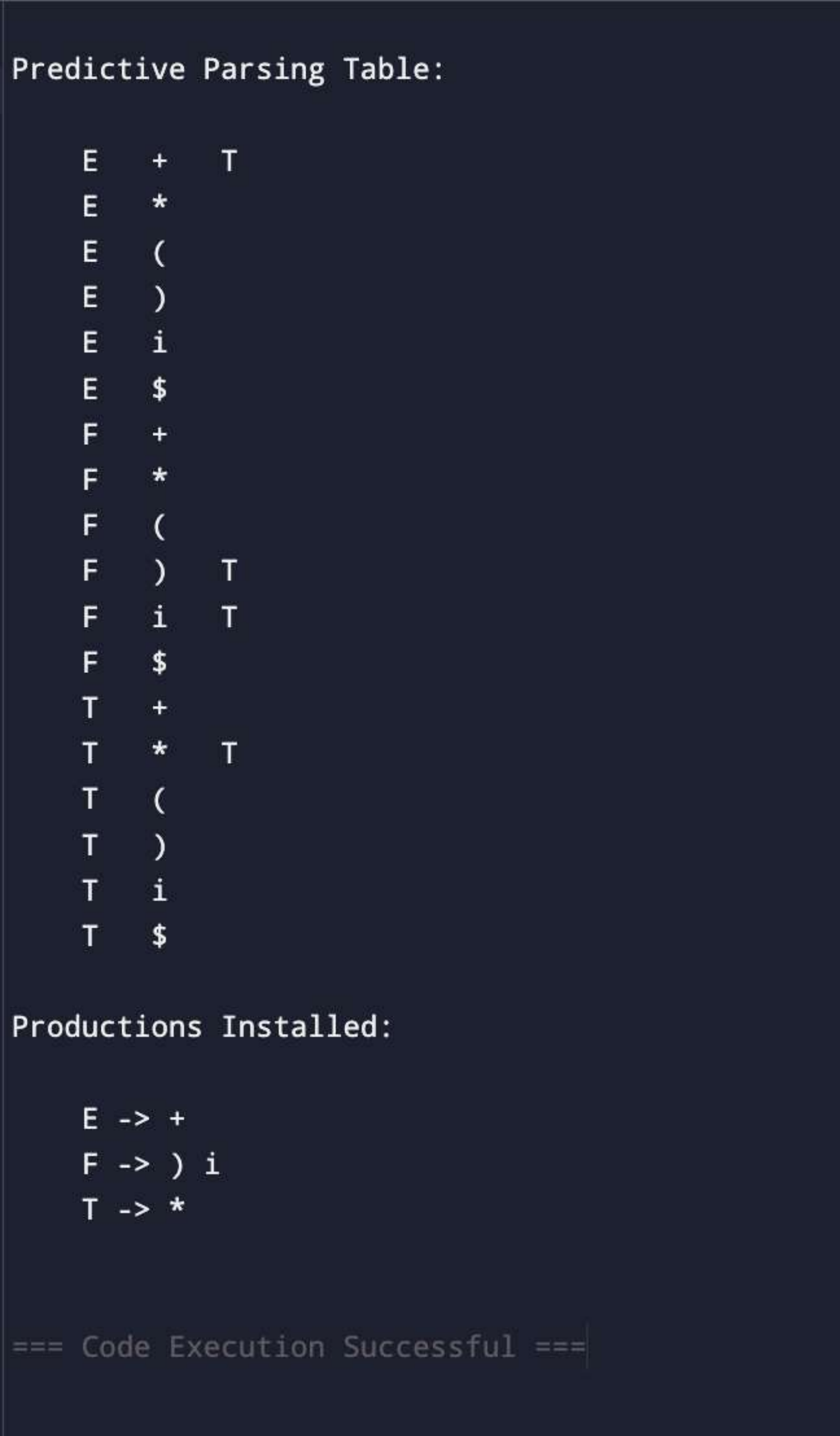


```c
#include <stdio.h>
#include <string.h>
char arr[18][3] = {
    {'E', '+', ' '}, {'E', '*', ' '}, {'E', '(', ' '}, {'E', ')', ' '},
    {'E', 'i', ' '}, {'E', '$', ' '}, {'F', '+', ' '}, {'F', '*', ' '},
    {'F', '(', ' '}, {'F', ')', ' '}, {'F', 'i', ' '}, {'F', '$', ' '},
    {'T', '+', ' '}, {'T', '*', ' '}, {'T', '(', ' '}, {'T', ')', ' '},
    {'T', 'i', ' '}, {'T', '$', ' '}
};
char prod[6] = "EETTFF";
char res[6][3] = {
    {'E', '+', 'T'},
    {'T', '\0', '\0'},
    {'T', '*', 'F'},
    {'F', '\0', '\0'},
    {'(', 'E', ')'},
    {'i', '\0', '\0'}
};
char stack[20][2];
int top = -1;
void install(char pro, char re) {
    int i;
    for (i = 0; i < 18; ++i) {
        if (arr[i][0] == pro && arr[i][1] == re) {
            arr[i][2] = 'T';
            ++top;
            stack[top][0] = pro;
            stack[top][1] = re;
            break;
        }
    }
}
```



```c
int main() {
    int i, j;
    char pro, re, pri = ' ';
    for (i = 0; i < 6; ++i) {
        for (j = 2; j >= 0; --j) {
            if (res[i][j] == '+' || res[i][j] == '*' || res[i][j] == '(' ||
                res[i][j] == ')' || res[i][j] == 'i' || res[i][j] == '$') {
                install(prod[i], res[i][j]);
                break;
            }
        }
    }
    printf("\nPredictive Parsing Table:\n");
    for (i = 0; i < 18; ++i) {
        printf("\n\t%c\t%c\t%c", arr[i][0], arr[i][1], arr[i][2]);
    }
    printf("\n\nProductions Installed:\n");
    for (i = 0; i < 18; ++i) {
        if (pri != arr[i][0]) {
            pri = arr[i][0];
            printf("\n\t%c -> ", pri);
        }
        if (arr[i][2] == 'T')
            printf("%c ", arr[i][1]);
    }
    printf("\n");
    return 0;
}

```



```
Predictive Parsing Table:

    E   +   T
    E   *
    E   (
    E   )
    E   i
    E   $
    F   +
    F   *
    F   (
    F   )   T
    F   i   T
    F   $
    T   +
    T   *   T
    T   (
    T   )
    T   i
    T   $

Productions Installed:

    E -> +
    F -> ) i
    T -> *


=== Code Execution Successful ===
```