College of Engineering Trivandrum

Compiler Design Lab

Abhishek Manoharan
S7 CSE Roll No:2

TVE17CS002

Department of Computer Science

November 13, 2020

Exp 13

1 Constant Propagation

1.1 Aim

Write a program to perform constant propagation.

1.2 Theory

Constant Propagation.

Expressions with constant operands can be evaluated at compile time, thus improving run-time performance and reducing code size by avoiding evaluation at compile-time. Constant propagation is the process of substituting the values of known constants in expressions at compile time. Such constants include those defined above, as well as intrinsic functions applied to constant values.

1.3 Algorithm

Algorithm 1: Algorithm for Constant propagation

```
Start
For all statement in the program do begin
for each output v of s do valout ( v , s )=unknown
for each input w of s do
if w is a variable then valin(w,s)=unknown
else valin(w, s )= constant value of w
end
```

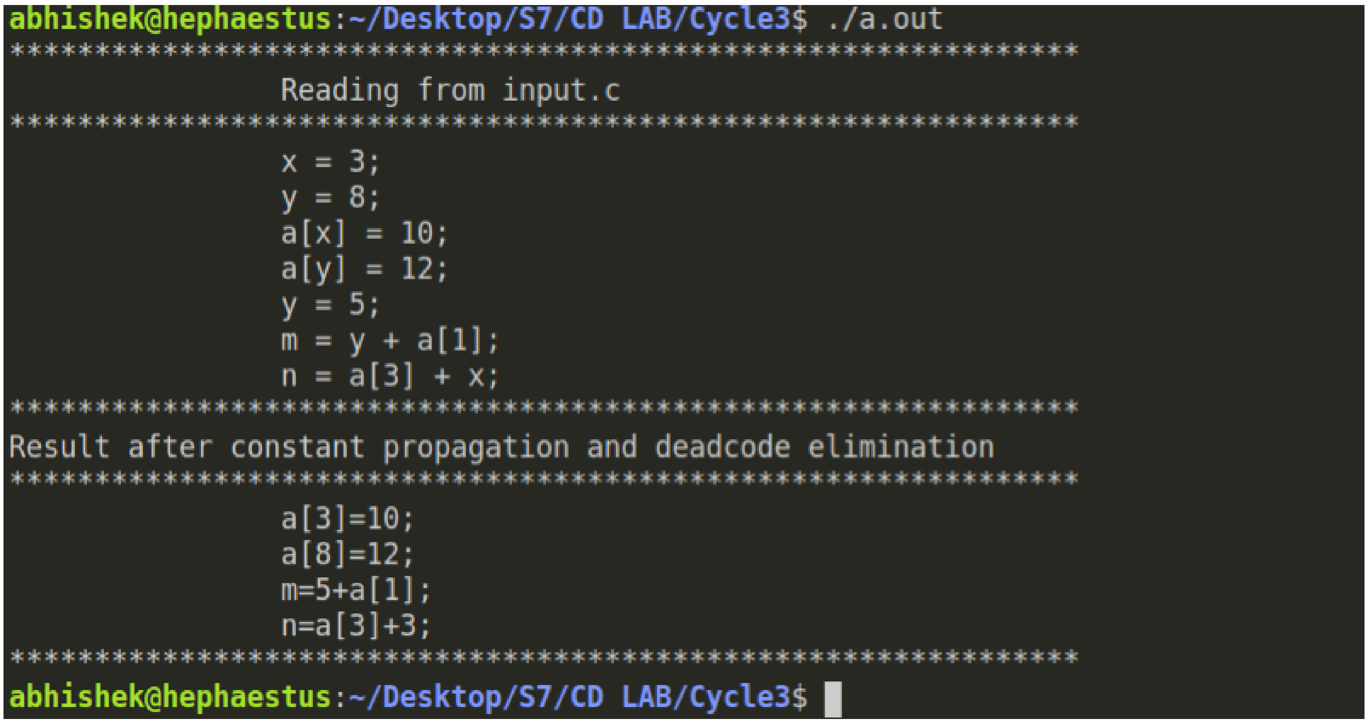
1.4 Code

```cpp
#include <bits/stdc++.h>
using namespace std;
string beautify(string s) // to remove unneccessery space , () etc in the loop
{
    string new_s = "";
    int n = s.size();
    int flag = 0;
    for (int i = 0; i < n; ++i)
    {
        if (s[i] != ' ')
        {
            new_s += s[i];
        }
    }
    return new_s;
}
void print_star()
{
    cout << "***************************************************************" << endl;
}
bool is_id(string s, int i)
{
    if (!isalpha(s[i]))
    {
        return false;
    }
    if (i == 0)
    {
        if (!isalnum(s[i + 1]))
        {
            return true;
        }
    }
    else if (i == s.size() - 1)
    {
        if (!isalnum(s[i - 1]))
        {
            return true;
        }
    }
    else
    {
        if (!isalnum(s[i - 1]) && !isalnum(s[i + 1]))
        {
            return true;
        }
    }
    return false;
}
vector<string> constant(vector<string> lines, unordered_map<char, int> values)
{
    vector<string> result;
    int n = lines.size();
    for (int i = 0; i < n; ++i)
    {
        int len = lines[i].size();
        if (regex_match(lines[i], regex("[a-zA-z]=[0-9]*;")))
        {
            //cout << "true" << endl;
            char variable = lines[i][0];
            string data = lines[i].substr(2, n - 1);
            int cons = stoi(data);
            //cout << "variale is: " << variable << " value: " << cons << endl;
            values[variable] = cons;
        }
        else
        {
            string append = "";
            for (int j = 0; j < len; ++j)
            {
                if (is_id(lines[i], j))
                {
                    if (values.find(lines[i][j]) != values.end())
                    {
                        int cons = values[lines[i][j]];
                        string s = to_string(cons);
                        append += s;
                        //cout << "variable found and appending " << s << endl;
                    }
                    else
                    {
                        append += lines[i][j];
                        // cout << "variable found but not value and appending " << lines[i][j
    ] << endl;
                    }
                }
                else
                {
                    append += lines[i][j];
                    //cout << "variable not found and appending " << lines[i][j] << endl;
                }
            }
            result.push_back(append);
            //cout << append << endl;
        }
    }
    return result;
}
int main()
{
    vector<string> lines;
    string s, temp;
    ifstream file("constant.c");
    print_star();
    cout << "\t\t"
         << "Reading from input.c" << endl;
    print_star();
    while (getline(file, s))
    {
        cout << "\t\t" << s << endl;
        s = beautify(s);
        lines.push_back(s);
    }
    unordered_map<char, int> values;
    vector<string> result = constant(lines, values);
    print_star();
    cout << "Result after constant propagation and deadcode elimination" << endl;
    print_star();
    for (auto x : result)
    {
        cout << "\t\t" << x << endl;
    }
    print_star();

    return 0;
}
```

Code for Constant Propagation

1.5 Output



```
abhishek@hephaestus:~/Desktop/S7/CD LAB/Cycle3$ ./a.out
***************************************************************
                Reading from input.c
***************************************************************
                x = 3;
                y = 8;
                a[x] = 10;
                a[y] = 12;
                y = 5;
                m = y + a[1];
                n = a[3] + x;
***************************************************************
Result after constant propagation and deadcode elimination
***************************************************************
                a[3]=10;
                a[8]=12;
                m=5+a[1];
                n=a[3]+3;
***************************************************************
abhishek@hephaestus:~/Desktop/S7/CD LAB/Cycle3$
```

1.6 Result

Implemented the program for constant propagation. It was compiled using g++ version 9.3.0, and executed in Ubuntu 20.04 and the above output was obtained.