МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Институт информационных технологий

Кафедра «Информационные технологии и компьютерные системы»

ОТЧЕТ

по лабораторной работе №8

по дисциплине “Системное программное обеспечение”

Вариант 4

Выполнил:
ст. гр. ИТ/б-22-6-о Донец Н.О.
Принял:
ассистент Ткаченко К.С.

Севастополь
2024 г.

Цель работы:

Изучить способы оптимизации объектного кода.

Задание:

Разработать процедуру оптимизации объектного кода, представленного в виде последовательности триад, способом удаления лишних переменных.

Ход работы:

Для выполнения поставленной в лабораторной работе задаче был разработан класс оптимизатора (листинг 1).

Листинг 1 – Оптимизатор

```
#ifndef OPTIMIZATOR
#define OPTIMIZATOR

#include <iostream>
#include <map>
#include <vector>

using namespace std;

class Optimizator {
    map<string, int> signOfUse;
    vector<vector<string>> objectCode;

    void ProcessOperation(vector<string>);
    bool ProcessAssignment(vector<string>);

    void ReplaceVariable(string);

    void IncreaseSignOfUse(string);

    string GetReplacingVariable(string var);

    public:
    vector<vector<string>> Optimize(vector<vector<string>>);
};

#endif
```

```
#include "Optimizator.h"

vector<vector<string>> Optimizator::Optimize(vector<vector<string>>
_objectCode) {
    objectCode = _objectCode;
    for (auto triad : objectCode) {
        if (triad[0] == ":=") {
            if(ProcessAssignment(triad)) {
                ReplaceVariable(triad[1]);
                signOfUse.erase(triad[1]);
            }
        }
        else {
            ProcessOperation(triad);
        }
    }
    return objectCode;
}

void Optimizator::ReplaceVariable(string replaceable) {
    cout<<"Replacing "<< replaceable <<endl;
    string replacing = GetReplacingVariable(replaceable);
    int flag = 0;
    auto toErase = objectCode.begin();
    for (auto triad = objectCode.begin(); triad != objectCode.end(); triad++)
{
        if (flag == 0) {
            if ((*triad)[1] == replaceable) {
                cout<<"Try erase"<<endl;
                toErase = triad;
                flag++;
                cout<<"Erased"<<endl;
            }
        }
        else if (flag == 1) {
            if ((*triad)[1] == replaceable) {
                (*triad)[1] = replacing;
                flag++;
            }
            else if ((*triad)[2] == replaceable) {
                (*triad)[2] = replacing;
                flag++;
                cout<<" Replaced by " << replacing << endl;
            }
        }
        if (flag != 0) {
            if ((*triad)[1][0] == 'T') {
                string str = "";
                for (int i = 0; i < (*triad)[1].size()-1; i++) {
                    str += (*triad)[1][i];
                }
                str += char(int((*triad)[1][(*triad)[1].size()-1]) - 1);
                (*triad)[1] = str;
            } else if ((*triad)[2][0] == 'T') {
                string str = "";
                for (int i = 0; i < (*triad)[2].size()-1; i++) {
                    str += (*triad)[2][i];
                }
                str += char(int((*triad)[2][(*triad)[2].size()-1]) - 1);
                (*triad)[2] = str;
            }
        }
    }
```

```
    objectCode.erase(toErase);
}

void Optimizator::ProcessOperation(vector<string> triad) {
    IncreaseSignOfUse(triad[1]);
    IncreaseSignOfUse(triad[2]);
}

bool Optimizator::ProcessAssignment(vector<string> triad) {
    if (signOfUse.find(triad[1]) != signOfUse.end()) {
        if (signOfUse[triad[1]] == 1) {
            IncreaseSignOfUse(triad[2]);
            return true;
        }
    }
    signOfUse[triad[1]] = 0;
    IncreaseSignOfUse(triad[2]);
    return false;
}

string Optimizator::GetReplacingVariable(string var) {
    cout<<"Findeing replacing value"<<endl;
    string replacing;
    auto triad = objectCode.begin();
    for ( ; (*(triad))[1] != var; triad++) {}
    replacing = (*triad)[2];
    cout<<"Founded: "<<replacing<<endl;
    return replacing;
}

void Optimizator::IncreaseSignOfUse(string var) {
    if (signOfUse.find(var) != signOfUse.end()) {
        if (signOfUse[var] != -1) {
            signOfUse[var]++;
        }
    }
    else {
        signOfUse[var] = -1;
    }
}
```

Также была разработана программа, проводящая тесты оптимизатора (листинг 2).

Листинг 2 – Тесты оптимизатора

```
#include "Optimizator.h"
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

vector<string> split(const string& str, const string& delim)
{
    vector<string> tokens;
    size_t pos = 0;
    while (pos < str.length())
    {
        // Find the position of the next occurrence of the delimiter
        size_t next = str.find(delim, pos);
```

```
        if (next == string::npos)
        {
            next = str.length();
        }
        // Extract the token from the string
        string token = str.substr(pos, next-pos);
        if (!token.empty())
        {
            tokens.push_back(token);
        }
        // Update the position to start searching from the next character
        pos = next + delim.length();
    }
    return tokens;
}

int main() {
    Optimizator optimizator;
    ifstream file("test.txt");
    vector<string> objCode;
    string str;
    while(getline(file, str)) {
        objCode.push_back(str);
        str.clear();
    }
    file.close();
    vector<vector<string>> objectCode;
    for (auto str : objCode) {
        objectCode.push_back(split(str, " "));
    }
    vector<vector<string>> optimized = optimizator.Optimize(objectCode);
    for (auto triad : optimized) {
        cout << setiosflags(ios::left) << setw(2) << triad[0] <<" " << setw(2)
<< triad[1] <<" " << setw(2) << triad[2] <<" " << endl;
    }
    return 0;
}
```

Был проведён тест оптимизатора (рисунок 1), для удобства проверки был использован пример из методических указаний (листинг 3). Ожидаемый результат представлен на листинге 4.

Листинг 3 – Объектный код

```
:= A  B
+  A  1
:= C  0
+  T2 C
+  C  T4
:= A  T5
```

Листинг 4 – Ожидаемый результат

```
+  B  1
:= C  0
+  T1 C
+  C  T3
:= A  T4
```

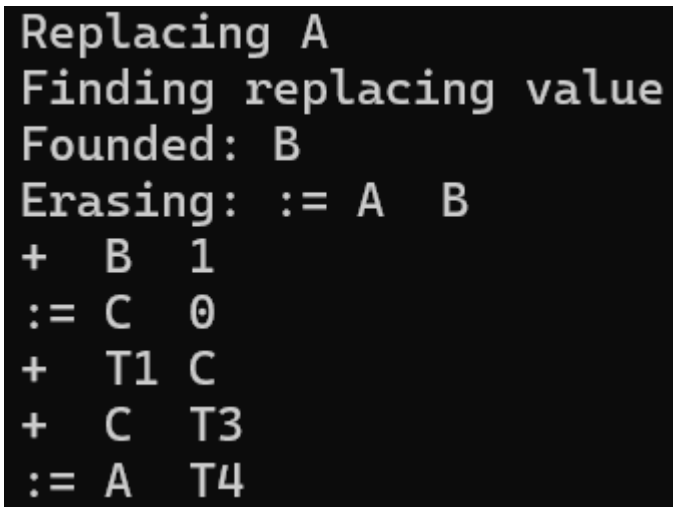

Рисунок 1 – Тест

Так как полученный результат (рисунок 1) совпал с результатом, ожидаемым в методических указаниях – можно сделать вывод, что оптимизатор работает корректно.

Выводы

В ходе лабораторной работы была изучена оптимизация способом удаления лишних переменных, а также была разработана и отлажена программа оптимизатора обозначенным ранее способом.