Лабораторная работа №3

Рефакторинг программного кода. Упрощение условных выражений

Цель работы

Исследовать эффективность рефакторинга программного кода путем упрощения условных выражений. Получить практические навыки применения приемов рефакторинга объектно-ориентированных программ.

Постановка задачи

1. Выбрать фрагмент программного кода для рефакторинга.

2. Выполнить рефакторинг программного кода, применив не менее 5 приемов, рассмотренных в разделе 2.2.

3. Составить отчет, содержащий подробное описание каждого модифицированного фрагмента программы и описание использованного метода рефакторинга.

Ход работы

1. Декомпозиция условного оператора (Decompose Conditional)

Код до рефакторинга:

```
    bool checkWin(char player)
    {
        bool isWin = false;
        for (int i = 0; i < 3; i++)
        {
            if (board[i][0] == player && board[i][1] == player && board[i][2] == player)
```

```
                {
                    isWin = true;
                }
                if (board[0][i] == player && board[1][i] == player &&
board[2][i] == player)
                {
                    isWin = true;
                }
            }
            if (board[0][0] == player && board[1][1] == player && board[2][2]
== player)
            {
                isWin = true;
            }
            if (board[0][2] == player && board[1][1] == player && board[2][0]
== player)
            {
                isWin = true;
            }
            return isWin;
        }
```

Код после рефакторинга:

```
    bool checkWin(char player)
        {
            return checkRows(player) || checkColumns(player) ||
checkDiagonals(player);
        }

        bool checkRows(char player)
        {
            for (int i = 0; i < 3; i++)
            {
                if (board[i][0] == player && board[i][1] == player &&
board[i][2] == player)
                {
                    return true;
                }
            }
            return false;
        }

        bool checkColumns(char player)
        {
            for (int i = 0; i < 3; i++)
            {
                if (board[0][i] == player && board[1][i] == player &&
board[2][i] == player)
                {
                    return true;
                }
            }
            return false;
        }
```

```
        bool checkDiagonals(char player)
        {
            return (board[0][0] == player && board[1][1] == player &&
board[2][2] == player) ||
                (board[0][2] == player && board[1][1] == player && board[2][0]
== player);
        }
```

2. Консолидация условного выражения (Consolidate Conditional Expression).

Код до рефакторинга:

```
int main() {
    AccessControl accessControl;

    accessControl.addUser("user", "qwerty123", Role::User);
    accessControl.addUser("moderator", "qwerty321", Role::Moderator);
    accessControl.addUser("admin", "123qwerty", Role::Admin);

    std::string name;
    std::string password;

    std::cout << "Enter your name: ";
    std::cin >> name;
    std::cout << "Enter your password: ";
    std::cin >> password;

    Role role = accessControl.login(name, password);

    if (role == Role::Guest) {
        std::cout << "Access denied.\n";
    } else if (role == Role::User) {
        std::cout << "Welcome, user.\n";
    } else if (role == Role::Moderator) {
        std::cout << "Welcome, moderator.\n";
    } else if (role == Role::Admin) {
        std::cout << "Welcome, admin.\n";
    }

    return 0;
}
```

Код после рефакторинга:

```
string getGreetingMessage(Role role) {
    static const unordered_map<Role, string> greetings = {
        {Role::Guest, "Access denied."},
        {Role::User, "Welcome, user."},
        {Role::Moderator, "Welcome, moderator."},
        {Role::Admin, "Welcome, admin."}
```

```
    };

    return greetings.at(role);
}

int main() {
    AccessControl accessControl;

    accessControl.addUser("user", "qwerty123", Role::User);
    accessControl.addUser("moderator", "qwerty321", Role::Moderator);
    accessControl.addUser("admin", "123qwerty", Role::Admin);

    string name;
    string password;

    cout << "Enter your name: ";
    cin >> name;
    cout << "Enter your password: ";
    cin >> password;

    Role role = accessControl.login(name, password);

    cout << getGreetingMessage(role) << '\n';

    return 0;
}
```

3. Консолидация дублирующихся (Consolidate Duplicate Conditional Fragments) условных фрагментов.

Код до рефакторинга:

```
if (age < 25)
    {
        if (carType == "sedan")
        {
            if (hasAccidents)
            {
                cost = 500.0;
            }
            else
            {
                cost = 1000.0;
            }
        }
        else if (carType == "suv")
        {
            if (hasAccidents)
            {
                cost = 750.0;
            }
            else
            {
```

```
            cost = 1500.0;
        }
    }
    else if (carType == "sports")
    {
        if (hasAccidents)
        {
            cost = 1000.0;
        }
        else
        {
            cost = 2000.0;
        }
    }
}
else if (age < 50)
{
    if (carType == "sedan")
    {
        if (hasAccidents)
        {
            cost = 250.0;
        }
        else
        {
            cost = 500.0;
        }
    }
    else if (carType == "suv")
    {
        if (hasAccidents)
        {
            cost = 375.0;
        }
        else
        {
            cost = 750.0;
        }
    }
    else if (carType == "sports")
    {
        if (hasAccidents)
        {
            cost = 500.0;
        }
        else
        {
            cost = 1000.0;
        }
    }
}
else
{
    if (carType == "sedan")
    {
        if (hasAccidents)
```

```
            {
                cost = 200.0;
            }
            else
            {
                cost = 400.0;
            }
        }
        else if (carType == "suv")
        {
            if (hasAccidents)
            {
                cost = 300.0;
            }
            else
            {
                cost = 600.0;
            }
        }
        else if (carType == "sports")
        {
            if (hasAccidents)
            {
                cost = 400.0;
            }
            else
            {
                cost = 800.0;
            }
        }
    }
```

Код после рефакторинга:

```
double baseCost = 0.0;
    double accidentSurcharge = 0.0;

    if (age < 25)
    {
        baseCost = 1.0;
    }
    else if (age < 50)
    {
        baseCost = 0.5;
    }
    else
    {
        baseCost = 0.4;
    }

    if (carType == "sedan")
    {
        baseCost *= 1000.0;
    }
    else if (carType == "suv")
```

```
    {
        baseCost *= 1500.0;
    }
    else if (carType == "sports")
    {
        baseCost *= 2000.0;
    }

    if (hasAccidents)
    {
        accidentSurcharge = 0.5;
    }

    return baseCost + (baseCost * accidentSurcharge);
```

4. Удаление управляющего флага (Remove Control Flag)

Код до рефакторинга:

```
#include <iostream>
#include <fstream>
#include <string>

int main() {
    std::ifstream file("text.txt");
    if (!file.is_open()) {
        std::cerr << "Failed to open file" << std::endl;
        return 1;
    }

    int lineCount = 0;
    int wordCount = 0;
    int charCount = 0;

    bool inWord = false;
    char c;
    while (file.get(c)) {
        charCount++;
        if (c == '\n') {
            lineCount++;
            inWord = false;
        } else if (std::isalpha(c)) {
            if (!inWord) {
                wordCount++;
                inWord = true;
            }
        } else {
            inWord = false;
        }
    }

    std::cout << "Line count: " << lineCount << std::endl;
    std::cout << "Word count: " << wordCount << std::endl;
    std::cout << "Char count: " << charCount << std::endl;
```

```
    file.close();
    return 0;
}
```

Код после рефакторинга:

```
#include <iostream>
#include <fstream>
#include <string>

int main() {
    std::ifstream file("text.txt");
    if (!file.is_open()) {
        std::cerr << "Failed to open file" << std::endl;
        return 1;
    }

    int lineCount = 0;
    int wordCount = 0;
    int charCount = 0;

    char prevC = ' ';
    char c;
    while (file.get(c)) {
        charCount++;
        if (c == '\n') {
            lineCount++;
        }
        if (std::isalpha(c) && !std::isalpha(prevC)) {
            wordCount++;
        }
        prevC = c;
    }

    std::cout << "Line count: " << lineCount << std::endl;
    std::cout << "Word count: " << wordCount << std::endl;
    std::cout << "Char count: " << charCount << std::endl;

    file.close();
    return 0;
}
```

5. Замена вложенных условных операторов граничным оператором (Replace Nested Conditional with Guard Clauses)

Код до рефакторинга:

Код после рефакторинга:

6. Замена условного оператора полиморфизмом (Replace Conditional with Polymorphism)

Код до рефакторинга:

Код после рефакторинга:

7. Введение объекта Null (Introduce Null Object)

Код до рефакторинга:

Код после рефакторинга:

ВЫВОДЫ

В ходе выполнения лабораторной работы была исследована эффективность рефакторинга программного кода путем упрощения условных выражений. Получены практические навыки применения приемов рефакторинга объектно-ориентированных программ.

Также были выбраны и отрефакторены участки кода программ, в которых были обнаружены проблемные условные выражения. В результате рефакторинга получившиеся условные выражения стали более понятными и читаемыми.