

**ПРОФЕСИОНАЛНА ГИМНАЗИЯ**

**ПО ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА “АПОСТОЛ АРНАУДОВ”**

гр. Русе, ул. “Потсдам” № 3; п.к. 7005, тел. 082/84-60-96; e-mail: [info-1806301@edu.mon.bg](mailto:info-1806301@edu.mon.bg)

Д И П Л О М Е Н П Р О Е К Т

**ЗА ПРИДОБИВАНЕ НА**

**ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ**

**по професия код 481020 „Системен програмист“**

**специалност код 4810201 „Системно програмиране“**

**ТЕМА:**

**SOLID принципите при разработка на софтуер в c#**

**Ученик**: Юлиян Велиславов Наумов

**Ръководител-консултант:** Милена Дамесова-Христова

гр. Русе

2024

**ГЛАВА ПЪРВА**

**Основи на езика C# и ООП концепцията**

1. **Основни възможности на програмния език C#**

C# (C Sharp, произнася се Си Шарп) е обектно ориентиран език за програмиране, разработен от Microsoft като част от софтуерната платформа .NET. Стремежът още при създаването на C# езика е бил да се създаде прост, модерен, обектно ориентиран език с общо предназначение. Основа за C# са C++, Java и донякъде езици като Delphi, VB.NET и C. Той е проектиран да балансира мощност (C++) с възможност за бързо разработване (Visual Basic и Java). Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика – инструкциите, които компютърът изпълнява. Програмите на C# представляват един или няколко файла с разширение .cs, в които се съдържат дефиниции на класове и други типове. Тези файлове се компилират от компилатора на C# до изпълним код и в резултат се получават асемблита – файлове със същото име, но с различно разширение (.exe или .dll).

Той е известен със своята синтаксисна яснота и гъвкавост, което го прави предпочитан избор сред програмистите по целия свят.

Сред основните възможности на C# са:

Обектно-ориентиран подход: C# поддържа основните концепции на обектно-ориентираното програмиране, като класове, обекти, наследяване, полиморфизъм и инкапсулация. Това позволява по-лесно управление на сложни проекти и по-ефективно използване на кода.

Мултиплатформеност: C# е разработен да бъде използван на различни операционни системи и платформи, включително Windows, Linux и macOS. Това осигурява гъвкавост и възможност за създаване на приложения, които могат да работят на различни устройства.

Интеграция с .NET Framework и .NET Core: C# е основен език за програмиране в рамките на .NET Framework и .NET Core, които предоставят обширен набор от библиотеки и инструменти за разработка на разнообразни приложения. Това включва уеб приложения, десктоп приложения, мобилни приложения и много други.

Съвременни функции на програмния език: C# постоянно се развива и включва съвременни функции и концепции на програмиране, като асинхронно програмиране, LINQ (Language Integrated Query), паттерн съпоставяне и др. Тези функции улесняват разработката на по-ефективен и чист код.

Богата екосистема от инструменти и ресурси: Заедно с C# и .NET идва богата екосистема от инструменти за разработка, IDE (Integrated Development Environment) като Visual Studio, както и голямо количество отворени библиотеки и ресурси за подпомагане на разработката.

Тези основни възможности на C# правят езика предпочитан избор за множество програмисти при създаването на разнообразни приложения.

* 1. **Типове данни и колекции**

**Прости типове**

Простите типове, който притежава C#, имат някои общи характеристики. Първо, всички те са псевдоними на .NET системните типове. Второ, изразите с константи от прост тип се изчисляват само при компилирането, не и при стартирането. На края, простите типове могат да бъдат инициализирани с литерали.

Простите типове в C# са групирани по следният начин:

 Целочислени типове

 Тип bool

 Тип char (специален случай на целичислен тип)

 Типове с плаваща запетая

 Типът decimal

* + 1. **Целочислени типове**

Съществуват девет целочислени типа в C#: sbyte, byte, short, ushort, int, uint, long, ulong и char (обсъден в отделна секция). Те имат следните характеристики:

 Типът sbyte представлява 8 битови цели числа със знак, чиито стойности са между -128 и 127.

 Типът byte представлява 8 битови цели числа без знак, чиито стойности са между 0 и 255.

 Типът short представлява 16 битови цели числа със знак, чиито стойности са между -32,768 и 32,767.

 Типът ushort представлява 16 битови цели числа без знак, чиито стойности са между 0 и 65,535.

 Типът int представлява 32 битови цели числа със знак, чиито стойности са между -2,147,483,648 и 2,147,483,647.

 Типът uint представлява 32 битови цели числа без знак, чиито стойности са между 0 и 4,294,967,295.

 Типът long представлява 64 битови цели числа със знак, чиито стойности са между-9,223,372,036,854,775,808 и 9,233,372,036,854,755,807.

 Типът ulong представлява 64 битови цели числа без знак, чиито стойности са между 0 и 18,446,744,073,709,551,615

Може би програмистите на C и VB ще бъдат малко изненадани от новите обхвати на типовете int и long. За разлика от други програмни езици, в C# типът int вече не е зависим от размера на думата за съответната система, а типът long е установен на 64 бита.

**1.1.1.1 Типът bool**

Типът bool представлява булевите променливи true и false. На булева променлива можете да присвоите една от двете стойности (true или false), или път да присвоите израз, чийто резултат се свежда отново до една от тях:

bool bTest = (100 > 90);

За разлика от C и C++, в C# стойността true вече не се представя от която и да била нулева стойност. Не съществува възможност за конвертиране на целочислени типове в булев такъв, за да се нарушава тази конвенция.

**1.1.1.2 Типът char**

Типът char представлява един Unicode символ. Unicode символът е със 16 бита дължина и може да бъде използван за представянето на повече езици по света. Можете да присвоите символ към променливата тип char по следният начин:

char chSomeChar = 'A';

В допълнение към това можете да присвоите стойност на символна променлива посредством шестнадесетична стойност с префикс x или Unicode формат с префикс u:

char chSomeChar = 'x0065';

char chSomeChar = 'u0065';

Не съществува безусловно конвертиране от тип char в някакъв друг тип данни. Това означава, че не можете да третирате променлива от този тип като някакъв друг целочислен тип в C# - това е още една подробност с която програмистите на C трябва да свикнат. Въпреки това, можете да извършите изрично преобразуване:

char chSomeChar = (char)65;

**1.1.1.3 Типове с плаваща запетая**

Два типа за данни спадат към категорията типове с плаваща запетая: float и double. Разликата между тях е в обхвата от сойности и броя цифри след десетичната запетая:

float: Обхватът от стойности е от 1.5 по 10 на -45 степен до 3.4 по 10 на 38 степен, с точност от 7 знака след десетичната точка.

double: Обхватът от стойности е от 5.0 по 10 на -324 степен до 1.7 по 10 на 308 степен с точност от 15-16 знака след десетичната точка.

При извършването на изчисления с които и да било от двата типа с плаваща запетая, могат да се получат следните стойности:

 Положителна или отрицателна нула

 Положителна или отрицателна безкрайност

 Не числена стойност (Not-a-Number - NaN)

 Крайно множество от не нулеви стойности

Друго правило при изчисленията на стойностите е, че ако една променлива в даден израз е от тип плаваща запетая, всички останали типове се конвертират в тип с плаваща запетая преди извършването на изчисленията.

**1.1.1.3 Типът double**

Типът decimal е много прецизен, 128 битов тип за данни, който е предназначен за употреба при финансови и парични изчисления. Той може да представя стойности, вариращи приблизително от 1.0 по 10 на -28 степен до 7.9 по 10 на 28 степен с 28 до 29 значими цифри. Важно е да се отбележи, че точността се определя от цифри, а не от десетични знаци. Операциите са точни, максимум до 28 десетични знака.

Както можете да видите, обхватът от стойности е по-малък от типа double, за сметка на това обаче е доста по-точен. Ето защо не е възможно безусловно конвертиране между типовете decimal и double - в едната посока можете да предизвикате препълване; в другата да загубите точността. Затова трябва изрично да заявите конвертиране посредством образец.

Когато дефинирате променлива и присвоявате стойност на нея, използвайте суфикса m, за да обозначите че стойността е от тип decimal:

decimal decMyValue = 1.0m;

**1.1.1.4 Типът int**

Типът данни int представлява цели числа в C#. Можете да присвоите стойности на променлива от тип int, като използвате цели числа без десетична запетая. При дефиниране на променлива от тип int, можете да използвате оператора за присвояване (=), за да й зададете стойност.

int age = 25;

int numberOfStudents = 50;

Целочислените типове данни в C# могат да представят както положителни, така и отрицателни числа, в рамките на определен обхват. В зависимост от конкретната имлементация, обхватът на типа int обикновено е от -2,147,483,648 до 2,147,483,647.

При използване на целочислени операции, трябва да внимавате за възможността за препълване или загуба на точност при извършването на операции, особено при деление или умножение на големи числа.

* 1. **Оператори**

A screenshot of a computer

Description automatically generated

Фигура 1 / Оператори /

**1.3 Управляващи инструкции**

**1.3.1 if-else case**

Условните конструкции if и if-else предоставят условен тип контрол, чрез който програмата може да изпълнява различни функции в зависимост от някакво условие, което се проверява по време на изпълнение на конструкцията.

public static void IfElse

{

if (булев израз)

{

тяло тип (1) на условната конструкция;

}

else if (нов булев израз)

{

тяло тип (2) на условната конструкция;

}

else

{

тяло тип (3) на else-конструкция;

}

}

Форматът включва: if клауза, булев израз и тяло на условната конструкция, if-else конструкцията се състои от: запазена дума if, булев израз, тяло на условната конструкция, запазена дума else, тяло на else конструкция. Тялото на else конструкцията може да се състои от един или няколко оператора, заградени в къдрави скоби, също както тялото на условната конструкция.

**1.3.2 switch case**

Конструкцията switch-case избира измежду части от програмен код на базата на стойност на зададения селектор, който представлява променлива или израз (най-често целочислен). Форматът на конструкцията за избор на вариант е следният:

public static void SwitchCase

{

switch (селектор)

{

case стойност на селектора (1): конструкция; break;

case стойност на селектора (2): конструкция; break;

default: конструкция; break;

}

}

Операторът switch сравнява резултата от селектора с всяка една стойност от изброените в тялото на switch конструкцията в case етикетите. Ако се открие съвпадение с някой case етикет, се изпълнява съответната конструкция. Ако не се открие съвпадение, се изпълнява default конструкцията.

**1.3.3 while loop**

While цикълът в C# е основна конструкция за повтаряне на определен блок код. Той се използва, за да повтаря определени действия, докато е изпълнено определено условие. Когато програмата достигне while цикъла, тя първо проверява условието. Ако това условие е верно, блокът от код в рамките на while цикъла се изпълнява. След като блокът приключи изпълнението си, програмата отново проверява условието. Ако то остане верно, цикълът се изпълнява отново. Този процес продължава, докато условието стане невярно, като по този начин while цикълът завършва.

public static void while

{

// Докато условието е вярно цикъла ще се изпълнява.

while (условие)

{

}

}

**1.3.4 for loop**

За разлика от while цикъла, for цикълът в C# изисква да бъде предварително определен брой итерации. Когато програмата достигне for цикъла, тя изпълнява инициализацията, след това проверява условието. След изпълнението на блока от код, програмата изпълнява стъпката за промяна и отново проверява условието. Този процес продължава, докато условието стане невярно, като по този начин завършва for цикълът.

public static void for

{

//Инициализация, следвана от проверка на условието, изпълнение на цикъла.

for (i=0; i<10; i++)

{

}

}

**1.3.5 foreach loop**

Foreach цикълът в C# се използва за итериране през елементите на колекция (като масив, списък и др.) без необходимост от указване на начало, край и стъпка за промяна. Вместо това, foreach цикълът автоматично преминава през всеки елемент от колекцията и изпълнява определен блок код за всяка итерация. Този вид цикъл е особено удобен, когато искаме да извършим действие върху всеки елемент на колекцията без да се налага да се грижим за индекси или броячи.

public static void foreach

{

//Изпълнение на цикъла за всеки елемент от масив или списък.

foreach (елемент in колекция)

{

}

}

1. **Етапи при разработката на софтуер**

Писането на софтуер може да бъде сложна задача, която отнема много време на цял екип от софтуерни инженери и други специалисти. Затова с времето са се обособили различни методики и практики, които улесняват живота на програмистите. Общото между всички тях е, че разработката на всеки софтуерен продукт преминава през няколко етапа, а именно: - Събиране на изискванията за продукта и изготвяне на задание; - Планиране и изготвяне на архитектура и дизайн; - Реализация (включва писането на програмен код); - Изпитания на продукта (тестове); - Внедряване и експлоатация; - Поддръжка. Фазите реализация, изпитания, внедряване и поддръжка се осъществяват в голямата си част с помощта на програмиране.