8 изпитен въпрос ПП-Разработка на софтуер

1. Описва предназначението и работата с дебъгер.

Процесът на откриване и поправяне или изолиране на грешки в компютърен програмен код се нарича дебъгване.

За дебъгване на програмата е необходимо да се:

Стартира програмата

Изолира източника на проблема

Поправи

Инструментите за отстраняване на грешки (наричани дебъгери) помагат за идентифициране на грешки в кода в различните етапи на разработка.

Отстраняване на грешки може да се разглежда като решаване на едно голямо дърво

Отделните възли представляват теоретични възможности

Отделните листа представляват възможните причини

Обхождането на дървото се свежда до изследване на състоянието

Минимизирането на времето за решение е ключово

Visual Studio IDE дава множество инструмента за дебъгване на приложението

-Добавяне на точки на прекъсване (стопери)

-Визуализиране на програмния поток

-Управление на потока на изпълнение

-Съвети

-Наблюдение на променливите

-Дебъгване на многонишкови програми

**Как да дебъгваме?**

Стартираме програмата с Visual Studio Debugger

Свързва се с вече стартиран процес.

Дебъгването е цялостно наблюдение на състоянието на процеса

Контролирането на изпълнението позволява :

Пауза в изпълнението

Възобновяване на изпълнението

Постъпково изпълнение на приложението на малки части

С помощта на IntelliTrace (записване на стъпките), се позволява връщане на стъпки назад и напред.

2.Коректно рефакторира и модифицира зле написан код, който нарушава добрите практики и стилови ръководства.

Какво ще рече преработка /рефакториране/на програмния код?

Подобряване на организацията и качеството на наличния програмен код, без да се променя неговото поведение

Това е постепенен процес, който превръща (ако е възможно) лошо написания код в добре написан

Защо бихме поискали преработка на кода?

-Програмният код непрекъснато се променя и от това качеството му се влошава (ако не се преработи)

-Изискванията често се променят и кодът трябва да се промени, за да отговори на новите изисквания

Като следвате основните правила и най-добрите практики за форматиране на C# код, преформатирайте дадените кодове и опишете правилата които са нарушени

Пример1

namespace YoloSnake.Interfaces {

public interface IDrawable {

void Draw(IDrawer drawer);

}

}

Пример2

namespace YoloSnake.Interfaces {

public interface IDrawer {

void DrawPoint(int

x, int

y, char

symbol

);

}

}

Пример3

namespace YoloSnake.Interfaces{

using System;public interface IKeyboardHandler{

ConsoleKey PressedKey

{

get;

}

bool

IsKeyAvailable

{ get; }

}

}

3.Описва и демонстрира създаването на компонентни тестове на предварително предоставен код.

Компонентът е код, явяващ се възможно най-малка част от приложението, която може да бъде самостоятелно тествана. В контекстът на C# това обикновено са методите.

Компонентен тест е код, който обикновено се пише от разработчика, с цел да се провери и гарантира правилното поведение на компонент.

Да се създаде компонентен тест на следната програма и се опишат извършените действия.

**int Sum(int[] array)**

**{**

**int sum = 0;**

**for (int i = 0; i < array.Length; i++)**

**sum += array[i];**

**return sum;**

**}**

4. Дефинира и различава понятията интегрирана среда за разработка (IDE) и тек- стов редактор

Интегрираните среди за разработак /Integrated Development Environments/ са софтуер предназначен за разработката на софтуер. Няма точно определени стандарти какво трябва да притежава всяко IDE, но обикновено:

Позволяват писане на код

Позволяват компилиране и дебъгване на код

Могат да имат модул за дизайн на интерфейси

Могат да бъдат свързани със системи за контрол на версиите

Могат да имат различни допълнителни модули и приставки

Разлики между текстов редактор и IDE

|  |  |  |
| --- | --- | --- |
|  | **Среди за разработка (IDE)** | **Текстови редактори** |
| Имат връзка с компилатори и дебъгери | Обикновено имат добра интеграция с компилатори и дебъгери | Обикновено поддържат редакция на код, но не и компилиране и дебъгване |
| Поддържка на програмни езици | Обикновено са специализирани с един или няколко конкретни програмни езика | Поддържат файлове от почти всички програмни езици, притежават модули за подчертване, оцветяване и автоматично допълване за голям набор от програмни езици |

5. Описва и демонстрира инструменти и клавишни комбинации за работа с код (бързо навигиране в кода, намиране на срещания и замяна, извличане – extract на методи и класове и др.) в интегрирана среда за разработка и текстов редактор.

Бързо стартиране /Quick launch/

Клавишна комбинация: CTRL + Q

Действие: помага ви бързо да намирате елементи от менюто, да изпълнявате команди и др.

Навигиране в кода /Navigate To/

Клавишна комбинация: CTRL + ,

Действие: помага ви бързо да търсите във файловете в рамките на solution-а

Навигиране в кода /Go to Definition/

Клавишна комбинация: F12 или десен бутон и Go To Definition

Действие: помага ви бързо да стигнете до дефиницията на даден клас/метод

Навигиране в кода /Peek Definition/

Клавишна комбинация: Alt+F12 или десен бутон и Peek Definition

Действие: показва ви прозорче с дефиницията под реда, на който сте селектирали

Кодови фрагменти /Code snippets/ - парченца код, които се появяват при написване на дадена дума и натискане на TAB

6. Описва и дава пример за употребата на инструменти в интегрирана среда за разработка и текстови редактори за работа със сорс-контрол система.

Система за контрол на версиите е механизмът, по който се управлява работата по даден софтуерен проект. За да се улесни разработката на софтуер са създадени специални системи, които намаляват неудобствата при съвместна работа на много хора върху един проект. Системите за контрол на версиите обикновено използват едно централно хранилище, в което се съхраняват файловете на проекта. Когато някой започне да работи по този проект, той създава копие на файловете от това хранилище на своята система и работи с тези копия. В процеса на работа потребителят може да изпраща към хранилището направените от него промени и да получава промените, направени по проекта от другите хора в екипа. След обновяване на хранилището, в него се запазват старите версии на променените файлове. Така тези файлове могат да бъдат възстановени при необходимост. Системата за контрол на версиите също така следи за конфликти – различни промени на различните потребители, които ползват хранилището.

Пример 1:-. Git: Основи на паралелна работа с GitЕмил 1.Създава

2.Модифицира изпраща на repository

Цвети 3.Клонира локално

4..Модифицира изпраща на repository

Пример 2: Разрешаване на конфликти със средствата на Git

Стефо - 1.променя файл изпраща на repository

Сашо 2.променя същия файл

3.Използва сливане (merge) или промяна на базата (rebase) - възниква

конфликт! 4. Разрешава конфликта и променя състоянието (commit + push)