VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

PROJEKTNA NALOGA

pri predmetu: Programiranje 1

Kandidat (-ka): »Tim Dolinsek«

Vrsta študija: študent (-ka) izrednega študija

Študijski program: informatika

Predavatelj:

Maribor, študijsko leto 2023/2024

IZJAVA O AVTORSTVU PROJEKTNE NALOGE

Podpisani/a »Tim Dolinsek«, sem avtor/ica projektne naloge.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel/a, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah – ZASP (Ur. l. RS 16/07 – uradno prečiščeno besedilo, 68/08, 110/2013, 56/2015 63/16, 59/19 in 130/22), prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo na spletnem portalu šole.

»Maribor«, »Junij, 2024«

Kazalo Vsebine

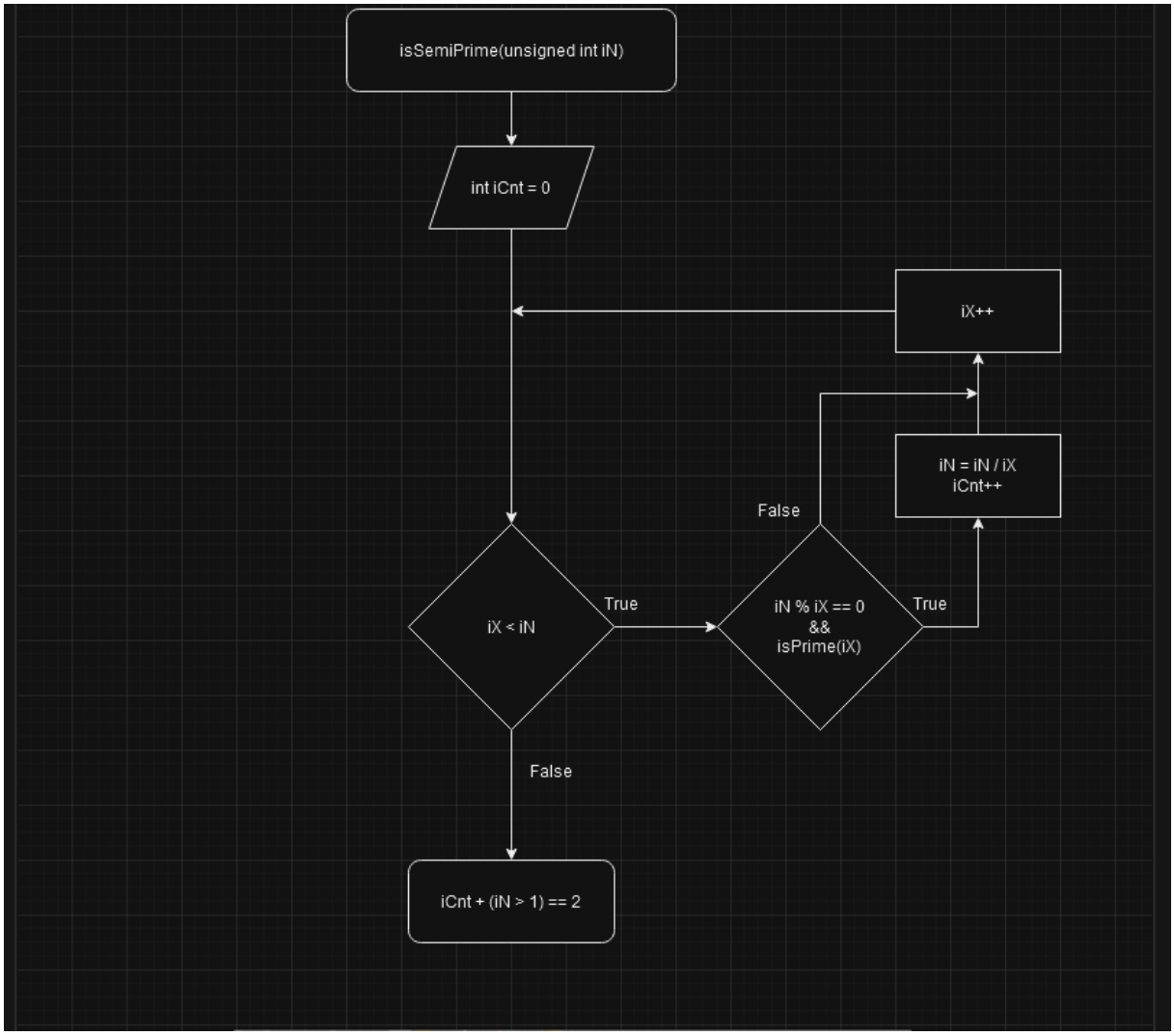
0.Github z projektom : https://github.com/DoliTim/primeNaloga

1. Opis postopka razhroščevanja pri točki 1

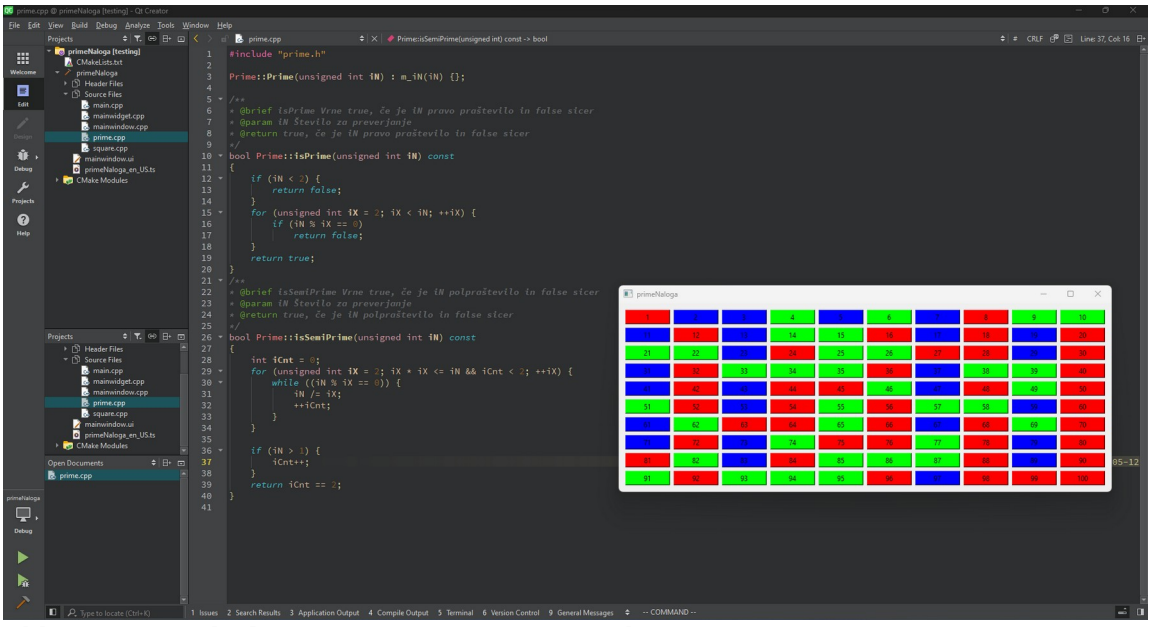
Pri razhroščevanju metod `isPrime` in `isSemiPrime` sem uporabil naslednje korake:

1. **Identifikacija napak**:
    - StackOverflow kjer sem primerjal kodo
    - Preveril sem logiko metode `isPrime`, kjer je bila napaka v preverjanju deliteljev.
    - V metodi `isSemiPrime` sem preveril logiko deljenja in števec deliteljev. Napaka je bila v logiki deljenja in pogojih za polpraštevilke.
2. **Uporaba razhroščevalnih orodij**:
    - Uporabili sem razhroščevalnik v IDE - Qt Creator, da sem korak za korakom pregledal izvajanje obeh metod.
    - Postavil sem točke prekinitve (breakpoint) pri ključnih pogojih in zankah, da sem spremljal vrednosti spremenljivk.
3. **Popravki kode**:
    - Metoda `isPrime` je bila popravljena, da preverja deljenje
    - V metodi `isSemiPrime` sem popravil logiko za deljenje s prvim pravim deliteljem in nato ponovnim deliteljem.

2. Diagram poteka iz točke 2



Slika 1 : Diagram poteka aplikacije Vir: (lasten vir)



Slika 2 :Primer kode Prime v Qt creatorju Vir: (lasten vir)

3. Primerjava postopkovne, objektne in dogodkovne paradigme

1. Postopkovna Paradigma

Postopkovna paradigma temelji na zaporedju ukazov, ki se izvajajo eden za drugim. Osredotoča se na funkcije in postopke za manipulacijo podatkov. **Funkcija `isPrime` je dober primer postopkovne kode, kjer je algoritem jasno določen s koraki.**

Prednosti:
- Enostavna za razumevanje in implementacijo za enostavne naloge.
- Primerna za algoritemske probleme.

Slabosti:
- Težko vzdrževanje in ponovna uporaba kode za večje in bolj kompleksne sisteme.
- Slaba modularnost, kar otežuje sodelovanje več razvijalcev.

2. Objektno Usmerjena Paradigma

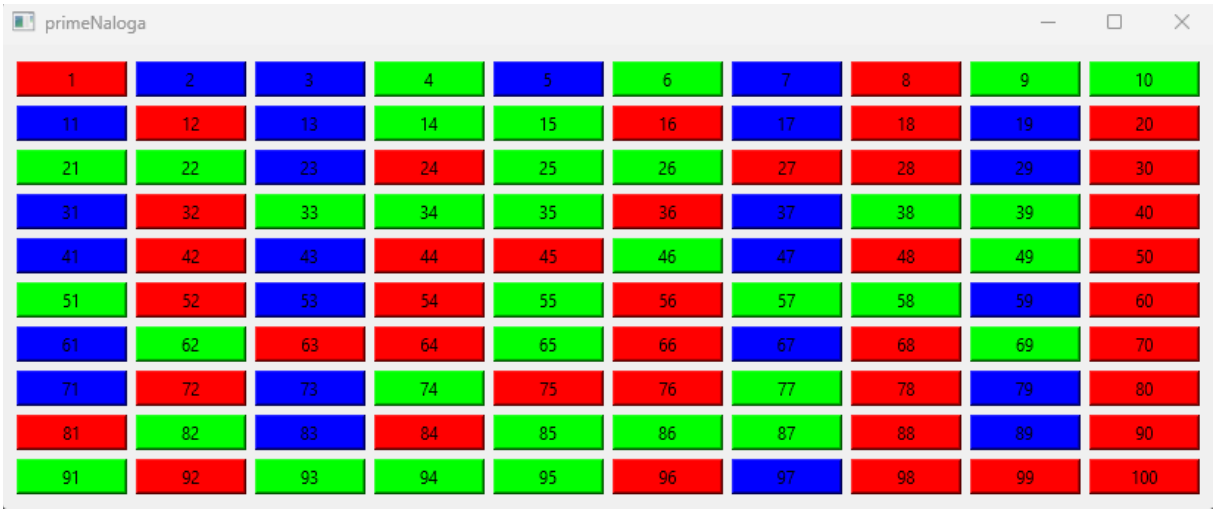
Objektno usmerjena paradigma temelji na konceptu objektov, ki združujejo podatke in metode. Uporablja razrede za definiranje objektov z lastnostmi in obnašanjem. **V nasem primeru razred `Prime`, ki vsebuje metode `isPrime` in `isSemiPrime`.**

Prednosti:
- Modularnost in ponovno uporabnost: Koda je razdeljena na majhne, samostojne enote (razrede), kar olajša ponovno uporabo in vzdrževanje.
- Enostavno modeliranje realnega sveta: Objekti lahko predstavljajo realne entitete z atributi in metodami.
- Boljša obvladljivost večjih projektov: Omogoča boljšo strukturo in organizacijo kode, kar je ključno pri večjih projektih.

Slabosti:
- Kompleksnost: Lahko je preveč kompleksna za manjše projekte in razvijalci potrebujejo več časa za razumevanje konceptov.

3. Dogodkovno Usmerjena Paradigma

Dogodkovno usmerjena paradigma temelji na dogodkih in odzivih na te dogodke. Pogosto se uporablja v razvoju grafičnih uporabniških vmesnikov (GUI) in interaktivnih aplikacij.



Slika 3 primer nase aplikacije Vir : (lasten vir)

Primer:
GUI aplikacija, ki označuje številke z različnimi barvami glede na to, ali so prava praštevila, polpraštevilka ali nič od tega.

Prednosti:
- Interaktivnost: Omogoča razvoj interaktivnih aplikacij, kjer so dejanja uporabnikov sprožilec za izvajanje kode.
- Ločitev logike in obdelave dogodkov: Jasno ločuje logiko aplikacije od obdelave dogodkov,kar omogoča boljšo organizacijo kode.

Slabosti:
- Kompleksnost upravljanja dogodkov: Večje število dogodkov lahko vodi do zapletene in težko sledljive kode.
- Odvisnost od ogrodja: Pogosto je močno odvisna od uporabljenega ogrodja (npr. Qt), kar lahko omejuje prenosljivost kode.

Sklepna Primerjava
- **Postopkovna paradigma je primerna za enostavne naloge in algoritemske probleme**, vendar se težje skalira in vzdržuje v večjih projektih.
- **Objektno usmerjena paradigma ponuja boljšo modularnost**, ponovno uporabnost in enostavno modeliranje realnega sveta, vendar je lahko prekompleksna za manjše projekte.
- **Dogodkovno usmerjena paradigma je idealna za interaktivne aplikacije in GUI**, vendar lahko postane zapletena zaradi velikega števila dogodkov in je pogosto odvisna od specifičnega ogrodja.

Vsaka paradigma ima svoje prednosti in slabosti, izbira pa je odvisna od specifičnih zahtev projekta.

4. Ocena vloge IDE-ja pri razvoju aplikacije

Qt Creator je zmogljivo integrirano razvojno okolje (IDE), ki je posebej zasnovano za razvoj aplikacij z uporabo Qt okvirja. V tem primeru smo ga uporabili za razvoj aplikacije, ki označuje prva 100 naravnih števil z različnimi barvami glede na to, ali so prava praštevila, polpraštevila ali nič od tega. Ključne funkcije Qt Creatorja, ki so bile uporabne pri tem projektu:

1. Učinkovito razhroščevanje

- **Postavljanje točk prekinitve (Breakpoints)**: Qt Creator omogoča enostavno postavljanje in upravljanje točk prekinitve, kar je bistveno pri prepoznavanju in odpravljanju napak v metodah `isPrime` in `isSemiPrime`.
- **Sledenje vrednostim spremenljivk**: Med izvajanjem aplikacije lahko spremljamo vrednosti spremenljivk v realnem času, kar omogoča natančno analizo poteka programa in hitro odpravljanje napak.

2. Napredno urejanje kode

- **Samodokončanje (Auto-completion)**: Qt Creator ponuja samodokončanje kode, kar bistveno pospeši pisanje in zmanjšuje možnost tipkarskih napak. To je še posebej koristno pri daljših imenih metod in razredov.
- **Predloge (Snippets)**: Uporaba predlog za pogosto uporabljene strukture kode pomaga pri doslednem in hitrem pisanju kode.

3. Vizualni oblikovalec za GUI (Qt Designer)

- **Oblikovanje uporabniškega vmesnika**: Qt Creator vključuje Qt Designer, ki omogoča vizualno načrtovanje uporabniškega vmesnika (GUI). To omogoča preprosto povleci in spusti oblikovanje, kar je bistveno za hitro postavitev vizualnih elementov aplikacije.
- **Takojšnja povratna informacija**: Vizualni oblikovalec omogoča takojšnjo povratno informacijo o spremembah, kar olajša prilagajanje in optimizacijo uporabniškega vmesnika.

4. Upravljanje projektov

- **Enostavno upravljanje datotek**: Qt Creator omogoča pregledno in strukturirano upravljanje datotek projekta. To vključuje enostavno dodajanje novih datotek, upravljanje virov in gradnjo projektov.
- **Večplatformska podpora**: Qt Creator podpira razvoj za različne platforme (Windows, macOS, Linux, mobilne naprave), kar omogoča pisanje prenosljive kode.

5. Integracija z različnimi orodji in knjižnicami

- **Podpora za različne jezike in knjižnice**: Qt Creator omogoča enostavno integracijo z različnimi knjižnicami in jeziki, kar omogoča uporabo dodatnih funkcionalnosti brez večjih težav.
- **Upravljanje različic (Version Control)**: Vgrajena podpora za sisteme za upravljanje različic (npr. Git) omogoča učinkovito sledenje spremembam v kodi, kar je ključno za sodelovanje v timskem okolju.

Qt Creator je celovito orodje, ki bistveno izboljšuje produktivnost pri razvoju aplikacij.