Индивидуални Пројекат 2023/24

Пројектни Захтеви

Увод

Честитамо Вам на успешном пласману у полуфинале квиза "Судокусфера"! Сада је време да се суочите са кључном игром која ће одредити Вашу судбину - "Судоку Изазов". У овом свету, бићете изазвани бројевима у судоку формату 3x3 квадрата.

На располагању је табла са 9 квадрата од по 9 ћелија. Правила су јасна - свака цифра од 1 до 9 мора се појавити тачно једном у:

- сваком реду читаве табле
- свакој колони читаве табле
- свакој од 3x3 подматрица читаве табеле

Да бисте могли да се упустите у решавање судоку задатка, неопходно је да напишете програм који ће Вам олакшати ово изазовно путовање кроз математички лавиринт. Припремите се за логичке подвиге, јер успех у игри "Судоку Изазов" води Вас корак ближе финалу "Судокусфере". Срећно!

Ток игре

- Покретање програма и менија за игру креираног у конзоли
- Креиран мени најпре пружа могућност одабира да ли корисник жели да учита већ постављену Судоку Загонетку из датотеке или жели да програм генерише нову
    - Уколико је одабрано генерисање поставке, програм креира датотеку у коју уписује поставку Судоку Загонетке кориснику на увид
- Кроз мени конзолне апликације корисник бира да ли жели да сам унесе решење учитавањем датотеке која садржи решену Загонетку или допушта програму да је реши
- На конзолној апликацији се након датог решења (генерисаног од стране програма или учитаног из датотеке корисника) приказују статистички подаци игре (број добро постављених поља, број погрешака, бројач одигране игре)
- Након одигране игре, корисник може да одабере понављање играња, чиме нова итерација игре започиње

Функционални захтеви

Написати Це++ програм који омогућава следеће:

1. **Учитавање аргумената командне линије**
    - Програм треба да рукује датотекама чији су називи дати кроз аргументе командне линије.
2. **Учитавање Судоку Загонетке:**
    - Програм треба да има функционалност за учитавање Судоку загонетке димензија 9x9 из датотеке у текстуалном формату.
    - За саму функционалност учитавања Загонетке у датотеку није важно да ли се учитава поставка Загонетке или њено решење, тј. није важно да ли је Загонетка која се учитава попуњена или не
3. **Чување Судоку Загонетке:**
    - Програм треба да исписује Судоку загонетку у датотеци.
    - Испис треба бити јасан и лак за разумевање.
    - За саму функционалност чувања Загонетке у датотеци није важно да ли је загонетка решена или не, важно је да јасан приказ матрице ради.
4. **Провера исправности Судоку решења играча:**
    - Програм треба да изврши проверу исправности решења који је играч унео кроз датотеку, осигуравајући да сваки број буде унесен само једанпут у сваком реду, колони и подматрици 3x3.
5. **Аутоматско Решавање Загонетке:**
    - Програм треба да има функционалност за аутоматско решавање Судоку загонетке.
    - Решење које понуди програм мора бити исправно, што значи да сва три правила игре Судоку морају бити испоштована.
6. **Поставка Судоку Загонетке:**
    - Програм треба да изврши генерисање валидне поставке једне Судоку загонетке 3x3 формата.
    - Просечан број постављених бројева, тј. попуњених поља по подматрици не сме да пређе 6.
7. **Испис статистичких информација**
    - Програм треба да након сваке партије изгенерише следеће информације:
        - Број унетих бројева који су на правом месту,
        - Број унетих бројева који су на погрешном месту,
        - Редни број партије која се игра.

Имплементациони кораци

Класа "Sudoku9" представља срж програма за рад са Судоку загонетком димензија 9x9. Садржи матрицу 9x9 за репрезентацију поља загонетке и основне функционалности. Поред ове класе потребно је по личном нахођењу креирати класе које дефинишу одређене логичке целине и смислено раздвајају функционалности. При изради пројекта разматрати коришћење концепата поменутих на вежбама попут полиморфизма, наслеђивања, композиције, инкапсулације и слично. У даљем тексту су дефинисани имплементациони кораци.

1. **Дефинисати класу за репрезентацију Судоку загонетке:**
    - Дефинишите класу Судоку која ће садржати матрицу 9x9 за репрезентацију Загонетке.
    - Размислите о додатним члановима класе који би могли бити корисни, попут бројача тачних уноса, бројача одигране игре, итд.
    - Размислите о модулима додатних чланова као и класа са којима класа Судоку интерагује, потребно је да програм буде модуларан.
2. **Имплементирати функционалност учитавања и исписивање Загонетке:**
    - Називи датотека из којих је могуће учитати или исписати Судоку Загонетку су задати кроз аргументе командне линије
    - Формат датотеке треба бити такав да сваки ред представља један ред Судоку загонетке, а сваки карактер у реду одговара вредности на том месту.
    - Приликом учитавања, уколико је Судоку Загонетка непопуњена, тј, уколико се учитава њена поставка, за поља која фале могуће је дефинисати идентификатор који то описује (нпр. број 0, број -1 и слично)
    - Приликом исписивања, идентификаторе који су означавали празна поља потребно је уклонити како би испис био уредан и јасан
    - Претпоставити да је улазна датотека дефинисана без грешака.
3. **Имплементирати функционалност- провере исправности уноса играча:**
    - Функције проверавају исправност матрице која представља решење Судоку Загонетке, без обзира на то да ли је ова матрица учитано решење од стране корисника или решење генерисано од стране програма
    - Програм ће пажљиво анализирати кориснички унос и утврдити да ли у њему има неправилности или су сва правила игре успешно испуњена
4. **Имплементирати алгоритме за решавање загонетке:**
    - На основу матрица која представља поставку, потребно је имплементирати функције које ће омогућити њено попуњавање поштујући правила игре Судоку

- Размислите о додавању оптимизација како бисте смањили време извршавања алгоритма.

5. **Имплементирати функционалност генерисања поставке Судоку Загонетке:**
   - Потребно је генерисати таблу Судоку Загонетке тако да бројеви и позиције на којима ће они бити генерисани буду псеудо-насумични, али да буду генерисани тако да обезбеде могућност правилног решавања Загонетке
   - За генерисање поставке и решавање Загонетке размотрите различите приступе: рекурзивни алгоритам, методу елиминације, комбинаторику и руковање пермутацијама, итд.
6. **Модуларност функционалности**
   - Све наведене функционалности могуће је имплементирати модуларно и независно једне од других. То треба користити приликом израде и тестирања решења како би сваки од корака био исправно валидиран и верификован
   - Водите рачуна о спрегама између функционалности (нпр. начину прослеђивања матрице) и о ефикасности кода (не правите непотребне копије, итд.)

Тестирање

За проверу функционалности програма, неопходно је написати додатне модуле у оквиру истог програма (или у одвојеном програму) који служе за његово тестирање. Модули за тестирање треба да буду искоришћени тако да позивају одговарајуће функције из модула решења које се тестирају. За сваку од функција која се тестира, модул за тестирање проверава да ли је њено понашање очекивано и на тај начин сигнализира да ли је тестни случај прошао или пао.

За потребе тестирања, није довољно користити искључиво тестне случајеве који ће очигледно радити, већ је потребно тестирати рад програма и са граничним случајевима у којима је функционалност програма упитна, како би тестови имали више смисла.

Стил кодирања

- У сваком заглављу и модулу са изворним кодом (раздвајати заглавља и датотеке са изворним кодом) додати кратак опис функционалности, информацију о ауторима, и датум и аутора последње измене.

- Коментарисати најбитније слободне функције, функције чланице и атрибуте класа. За функције обезбедити кратак опис функционалности, листу улазних аргумената и повратну вредност.
- Обратити пажњу на увлачење линија и форматирање кода, на стил и формат именовања променљивих и функција, као и на дужине линија.
- Код треба да буде прегледан, читљив и да садржи корисне коментаре.

Извештај

Треба да садржи следеће:

- Насловну страну са информацијама о аутору.
- Опис и/или анализу:
  - Рада У/И подсистема (учитавање и испис/упис).
  - Списак свих класа, изузетака и слободних функција.
  - Објашњење најбитнијих атрибута, класа и функција чланица, слободних функција и изузетака.
  - Структуре аргумената командне линије и пример коришћења.
  - Структуре улазне и излазне датотеке.
  - Опис алгоритма за решавање загонетке.
  - Опис начина тестирања.
  - Уочени проблеми и ограничења.
- Пример структуре добре документације објављен је уз пројекат (имајте у виду да је неопходно прилагодити називе поглавља и њихов садржај)

Упутство за предају пројекта

Пројекат треба да буде архивиран и именован на следећи начин:

Indeks_Ime_Prezime.zip
(пример: SW123_Petar_Petrovic.zip)

- Архиву поставити на КАНВАС у предвиђеном временском року истакнутом кроз обавештење на КАНВАС платформи.
- Архива треба да садржи следеће директоријуме:
  - **Кодови** – садржи датотеке са изворним кодом (.cpp и .h/.hpp) и пројектне датотеке (.sln, vcxproj и filters).
  - **Тестови** – садржи све тестне датотеке.
  - **Документација** – садржи пројектну документацију **Izvestaj.pdf**.

- Архива **НЕ СМЕ** садржати следеће датотеке:
  - .sdf
  - .suo
  - .user
  - .obj
  - .lib
  - .exe
  - ipch
  - Debug директоријум
  - Release директоријум

На термин одбране потребно је **донети штампану верзију документације**.

НАПОМЕНЕ:

- У циљу квалитетније провере рада програма користити више тестних датотека са различитим бројевима.