Индивидуални пројекат

Последњи рок за предају пројекта: 11.01.2023. 23:59

Пројектни захтеви

Увод

Честитамо! Успели сте да се пласирате у полуфинале квиза „Слагалица“. Како бисте прошли даље, одлучујућа игра је „Мој број“. Време је ограничено, а противник брз.-

Пронађите решење за учитан троцифрен број са улаза: на располагању имате 6 насумично генерисаних бројева (4 из скупа једноцифрених бројева и 2 из скупа двоцифрених бројева), операције: +,-,*,/, као и заграде. Резултат мора бити тачан, или уколико то није могуће, што ближи задатом троцифреном броју.

Али, да бисте уопште могли да одиграте игру, прво морате да напишете програм који вам то омогућава.

Функционални захтеви

Написати Це++ програм који омогућује следеће:

- Учитавање из улазне текстуалне датотеке поставке задатка (6 бројева и један тражени број), једну по једну. Свака поставка представља једну рунду. Сваки непарни унос прво решава ИГРАЧ А, а сваки парни – ИГРАЧ Б. Ако први играч добије тачан број, освојио је рунду и други играч ни не решава. У супротном, други играч има шансу да добије тачан број, или број ближи тачном. (Разлика у односу на варијанту на телевизији је у томе што нема лицитирања на почетку.) Након свих рунди (до краја датотеке), победу односи играч који је освојио више рунди.

- Срачунава вредност израза које играчи уносе, ради потребне провере и утврђује да ли је добијен тачан број, то јест колико се добијени број разликује од траженог. (У случају неисправно унетог израза тражи понављање уноса)

- Исписује најбоље могуће решење након сваке рунде. За решењем трага тако што проверава све могуће изразе који се могу саставити од задатог улаза.

- Исписује у излазну датотеку „Rezultati.txt“:

    - Број рунде.

    - Податке о одиграној рунди: бројеве који су били на располагању за рачунање и тражени троцифрени број; троцифрени број који је добио сваки од играча у тој рунди, колико он одступа од траженог, као и израз који је дово до тих бројева; информацију који играч је добио рунду; и на послетку – решење које сам програм пронашао;

    - На крају датоеке, исписује број освојених рунди оба играча и закључује који играч је укупни победник.
- Подржати прослеђивање неопходних улазних параметара програма (назив улазне датотеке) преко аргумената командне линије.

Имплементациони кораци

1. Програм не рачуна задати израз, већ га само памти. Унос израза може привремено да буде у форми: ***резултат=израз***. Програм онда само издвоји ***резултат*** и то узима у обзир приликом вођења игре (ко је добио тачан број, ко је ближи итд.). Решење које је сам програм нашао и излазној датотеци треба да буде само текст „tacan_izraz“, или нешто слично. У овом кораку нагласак је да се улаз/излаз разради.
2. Програм рачуна задате изразе (унос ираза се сада враћа у нормалну форму). И даље не налази решење самостално. У овом кораку игра је већ игрива.
   2.1. На улазу се увек претпостављају ваљани изрази, тј. нема опоравка од погрешног

уноса израза од стране корисника/такмичара.

2.2. У случају лошег израза на улазу програм детектује проблем и поново затражује унос.

3. Програм самостално налази решење, које предочава такмичарима на крају сваке рунде.
   3.1. Испрабавају се само изрази у којима учествује свих 6 бројева. Заграде се не узимају у обзир. То је 6! пермутација бројева, пута 4&5 комбинација операција (4 операције, и 5 места између 6 операнада). Пошто се не користе заграде и не гледају изрази са мање од 6 бројева, најбоље решење можда неће бити пронађено за сваки улаз.
   3.2. Заграде се сада узимају у обзир. Ефекат заграда се може опонашати тако што се уведу две додатне операције које имају исти смисао као + и -, само са приоритетом већим од * и /. Број комбинација операција тако расте на 6&5. Међутим, испис нађеног израза треба да буде са заградама. (Обратити пажњу да се на овај начин не проверава сваки могући израз са заградама, али ће се проверити израз исте вредност. На пример, неће се проверити 5 / (6 / 7), али то је исто што и 5 * 7 / 6, што ће бити проверено.)
   3.3. Прођи и кроз изразе који имају мање од 6 бројева.
4. Написати део кода који рачуна вредност израза генерички, тј. као шаблон, тако да може да ради са произвољним типом операнада у изразу.

НАПОМЕНЕ:

- Ослоните се на стандардну библиотеку. Не заборавите на заглавље <algorithm>.
- Део кода који срачунава израз можете искористи у делу кода који тражи најбоље решење.

Тестирање

Осим ручне провере програма, потребно је написати и посебни мали програм који ће на основу задатих тестних случајева проверавати ваљаност модула за срачунавање израз. Модул за проналажење решења треба такође проверити у посебном програму, тако што ће изгенерисано решење за задат улаз слати модулу за срачунавање израза, а онад резултат проверавати са очекиваним.

Захтеви улаза и излаза (У/И)

Потребно је омогућити следећу У/И спрегу са корисником:

- Прослеђивање потребних аргумената преко командне линије (назив датотеке).
- Учитавање података из улазне датотеке.
- Обезбедити испис и чување:
  - Крајњих резултата

    - Података прикупљених у току игре, за сваку рунду
- Претпоставити да је улазна датотека дефинисана без грешака.
- Произвољно дефинисати формат и структуру излазне датотеке.

Стил кодовања

- 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, повратну вредност и уколико користи, тип изузетка.
- Обратити пажњу на увлачење линија и форматирање кода, на стил и формат именовања промељивих и функција, као и на дужине линија.
- Код треба да буде прегледан, читљив и да садржи корисне коментаре.
- Више о начину и стилу кодовања доступно је у следећем документу: http://www.stroustrup.com/Programming/PPP-style-rev3.pdf

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, изузатака и слободних функција.
  - Објашњење најбитних атрибута, класа и функција чланица, слободних функција и изузетака.
  - Структуре аргумената командне линије и пример коришћења.
  - Структуре улазне и излазне датотеке.
  - Опис алгоритма за проналажење тачног решења.
  - Опис начина тестирања.
  - Уочени проблеми и ограничења.

Упутство за предају пројекта

- Пројекат треба да буде архивиран и именован на следећи начин:

Indeks_Ime_Prezime.zip
(пример: SW123_Petar_Petrovic.zip)

- Архиву послати предметном асистенту у предвиђеном временском року.
- Архива треба да садржи следеће директоријуме:
  - **Kodovi** – садржи датотеке са изворним кодом (.cpp и .h/.hpp) и пројектне датотеке (.sln, vcxproj и filters).
  - **Testovi** – садржи све тестне датотеке.
  - **Dokumentacija** – садржи пројектну документацију *Izvestaj.doc*.
- Архива **НЕ СМЕ** садржати следеће датотеке:
  - .sdf
  - .suo
  - .user
  - .obj
  - .lib
  - .exe
  - ipch
  - Debug директоријум
  - Release директоријум
- Термини одбране ће бити организовани у недељи од 15.01. до 19.01. За тачне термине по групама, пратите сајт Одсека.
- На термин одбране потребно је **донети штампану верзију документације**.

НАПОМЕНЕ:

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