Programowanie i struktury danych – egzamin

Organizacja plików: *Wszystkie pliki oddawane do sprawdzenia należy zapisać we wspólnym folderze o nazwie będącej numerem indeksu, umieszczonym na pulpicie.* ***Oddajemy tylko źródła programów (pliki o rozszerzeniach .cpp i .h)!!!***

Napisać pomocnicze biblioteki oraz program główny dla następującego zadania:

Centralna Policja Podatkowa raz w roku kontroluje zeznania podatkowe obywateli Fiskustanu. Lista podatników nazywa się "podatnicy.txt" i zawiera ich dane uporządkowane rosnąco wg nazwiska i dalej imienia w formacie (kolejne linie):

```
NIP (10-cyfrowy numer identyfikacji podatkowej)
Nazwisko Imię (oddzielone odstępem)
<pusta linia>
```

Plik z rozliczeniami podatników nazywa się "rozliczenia.txt" i zawiera liczbę linii zgodną z liczbą podatników. Każda linia zawiera kwotę rozliczenia odpowiadającego jej podatnika zapisaną z dokładnością do 2 miejsc dziesiętnych, np.:

```
195.67
0.00
-312.50
```

Kwoty rozliczeń mogą być liczbami dodatnimi (niedopłaty podatku), ujemnymi (nadpłaty) lub zerami (podatek rozliczony). Należy wydrukować do pliku "wyniki.txt" listę zawierającą dane podatników wraz z odpowiadającymi im kwotami rozliczenia, uporządkowaną malejąco wg nazwiska i dalej imienia podatnika, w następującym formacie (kolejne linie):

```
Nazwisko
Imię
NIP (numer identyfikacji podatkowej)
Kwota rozliczenia (zaokrąglona do liczby całkowitej)
<pusta linia>
```

Wymagania formalne:

Program powinien być zapisany w trzech plikach: kontener.h, dane.h i glowny.cpp (wolno rozbić kontener na kontener.h i kontener.cpp, podobnie z dane.h i dane.cpp).

W pliku dane.h należy umieścić strukturę implementującą dane przetwarzane w programie, przechowywane w kontenerze, wraz z odpowiednimi metodami. Tu powinny znaleźć się też wszystkie pomocnicze funkcje, sprawdzające poprawność danych (na przykład poprawność numeru NIP). Nie należy umieszczać w tym pliku funkcji formatujących dane do wydruku, ani żadnych innych funkcji używających operacji wejścia-wyjścia.

W pliku kontener.h należy umieścić strukturę odpowiadającą użytemu abstrakcyjnemu typowi danych. Można użyć szablonu lub konkretnej struktury z typem elementu odpowiadającym strukturze zbudowanej w pliku dane.h. Należy umieścić w kontenerze i implementować tylko te metody, które będą używane w programie głównym. Implementacja nie może zależeć od typu elementu, powinna działać tak samo po zmianie tego typu na jakikolwiek inny. Nie może też być tu żadnych operacji wejścia wyjścia ani formatowania wydruku kontenera.

W pliku glowny.cpp umieszczamy program główny korzystający z plików kontener.h i dane.h oraz funkcje formatujące wydruk i inne funkcje korzystające z operacji wejścia-wyjścia.

Niedozwolone jest korzystanie z kontenerów i algorytmów z biblioteki standardowej C++. Zapisu do pliku wynikowego nie wolno rozpocząć przed odczytaniem całości plików wejściowych.

Dodatkowe punkty:

Za obsługę wejścia odporną na błędy użytkownika i za poprawne komentarze, w tym w bibliotekach pomocniczych: cel funkcji, warunki wstępne, warunki końcowe, sytuacje wyjątkowe i ewentualnie zwracany wynik, przyznawane będą dodatkowe punkty. Obsługa wejścia odporna na błędy użytkownika powinna wyglądać następująco:

- w przypadku gdy niepoprawne są dane podatnika (czyli nieprawidłowość wystąpiła w pliku podatnicy.txt), podatnik ten nie jest wpisywany do plik wynikowego, a odpowiadająca mu linia w pliku rozliczenia.txt jest pomijana,
- w przypadku gdy dane podatnika są poprawne, a niepoprawna jest kwota rozliczenia (czyli nieprawidłowość wystąpiła w pliku rozliczenia.txt), podatnik jest wpisywany do pliku wynikowego, przy czym zamiast kwoty rozliczenia pojawia się adnotacja „DO WERYFIKACJI".

Kolejne punkty podwyższające ocenę można uzyskać zapewniając w programie obsługę nieograniczonej liczby podatników zapisanych w plikach wejściowych oraz obsługę wielu imion podatników podanych z odstępami w linii za nazwiskiem.

Dodatkowe punkty będą również przyznane za weryfikację poprawności numeru NIP. Ostatnia, dziesiąta cyfra NIP jest cyfrą kontrolną obliczaną według poniższego algorytmu:

1. Pomnożyć każdą z pierwszych dziewięciu cyfr odpowiednio przez wagi: 6, 5, 7, 2, 3, 4, 5, 6, 7.
2. Zsumować wyniki mnożenia.
3. Obliczyć resztę z dzielenia przez 11 – powinna być ona zgodna z cyfrą kontrolną.

NIP jest tak generowany, aby nigdy w wyniku tego dzielenia nie wyszła liczba 10.