Zadanie D - Podprogramy

Punktów procentowych do uzyskania: **10**

Język programowania: C++

autor zadania: Marek Śmieja

Opis zadania

Zadanie wymaga implementacji:

1. Funkcji typu `int` o nazwie `Suma` zwracającej sumę początkowych elementów tablicy typu `int`. Pierwszy argument jest wartością typu `int` przekazującą ilość elementów do sumowania, natomiast drugi argument przekazuje nazwę tablicy.

2. Funkcji logicznej o nazwie `SredniaWazona` z pierwszym argumentem służącym przekazaniu tablicy elementów typu `double`, drugim argumentem typu `int` określającym w opisany niżej sposób liczbę używanych elementów tablicy przekazanej pierwszym argumentem i trzecim argumentem typu `double` służącym przekazaniu poprzez referencję wartości wyliczonej przez funkcję w opisany dalej sposób.

   Zakładając, że drugi argument przekazuje wartość n, tablica przekazana pierwszym argumentem zawiera liczby:

   $$a_1, p_1, a_2, p_2, ..., a_{n-1}, p_{n-1}, a_n$$

   a ponadto spełnione są warunki:
   - $p_i \geqslant 0$, dla $i = 1, 2, ..., n - 1$
   - $p_i \leqslant 1$, dla $i = 1, 2, ..., n - 1$
   - $p_1 + p_2 + ... + p_{n-1} \leqslant 1$

   Jeżeli którykolwiek z podanych wyżej warunków nie jest spełniony, to funkcja zwraca wartość `false` nie zmieniając wartości żadnego z argumentów. Natomiast jeżeli powyższe warunki są spełnione, to funkcja zwraca wartość `true` oraz do trzeciego argumentu przekazuje średnią ważoną danych opisanych pierwszym argumentem według wzoru:

   $$p_1 a_1 + p_2 a_2 + ... + (1 - p_1 - p_2 - ... - p_{n-1}) a_n$$

3. Funkcji typu `short` o nazwie `PierwiastkiKwadratowe` przewidującej trzy pierwsze argumenty typu `double` służące przekazaniu wartości kolejnych współczynników trójmianu kwadratowego począwszy od współczynnika przy najwyższej potędze. Funkcja powinna zwrócić ilość różnych pierwiastków trójmianu kwadratowego o współczynnikach zadanych pierwszymi trzema argumentami, podczas gdy dwa kolejne argumenty typu `double` służą przekazaniu przez referencję wartości ewentualnych pierwiastków, co również zadaniem omawianej funkcji.

   Zarazem w przypadku:
   - Dwóch różnych pierwiastków ostatnie argumenty przejmują ich wartości w kolejności koniecznie rosnącej.
   - Jednego pierwiastka jego wartość jest przekazywana do przedostatniego argumentu pozostawiając ostatni argument bez zmian.
   - Braku pierwiastków dwa ostatnie argumenty nie są zmieniane.

4. Procedury `Fibonacci` z pierwszym argumentem przekazującym tablicę elementów typu `int` oraz drugim argumentem typu `int` będącym wartością indeksu ostatniego elementu tablicy podlegającego działaniu procedury. Po wykonaniu procedury wszystkie elementy tablicy danej pierwszym argumentem począwszy od elementu o indeksie 0 aż do elementu danego drugim argumentem włącznie muszą zawierać kolejne wyrazy ciągu Fibonacciego, zaś pozostałe elementy tablicy muszą pozostać bez zmian.

5. Funkcji typu `int` o nazwie `Licznik` z pierwszym argumentem będącym wartością typu logicznego oraz drugim argument typu `int` służącym przekazaniu przez referencję opisanej dalej wartości. Funkcja powinna zwracać ilość swoich dotychczasowych wywołań z pierwszym parametrem o wartości `false`, zaś w drugi argumencie przekazywać dotychczasową ilość wywołań z pierwszym parametrem wynoszącym `true`.

Dodatkowe uwarunkowania

- Całość rozwiązania musi znaleźć się w pliku o nazwie `kod.cpp` i obejmować WYŁĄCZNIE implementację wymaganych podprogramów.
- W szczególności plik `kod.cpp` nie może zawierając funkcji `main`.
- Wysyłany na BaCę plik z rozwiązaniem MUSI być skompresowany programem `zip`.
- Jedynym plikiem nagłówkowym dopuszczonym do włączenia jest plik `cmath`.
- **Zabronione jest używanie rekurencji.**
- **Zabronione jest używanie zmiennych globalnych.**

Przykłady użycia wymaganych podprogramów

Dla przykładowego kodu głównego programu:

```cpp
#include <iostream>
using namespace std;

#include "kod.cpp"

int main()
{
 int tab [] = { 1, -2, 3, -4, 5, -6 };
 int n = 6;

 cout << Suma ( n, tab ) << endl;
 tab [ 1 ] = tab [ 1 ] * ( -1 );
 cout << Suma ( n, tab ) << endl;

 double c = 1;
 double wagi [] = { 0.1, 0.3, 0.15, 0.05, 0.2 };
 double srednia [ 2 * n - 1 ];
 for ( int i = 0; i < n - 1; ++i ) {
     srednia [ 2 * i ] = tab [ i ];
     srednia [ 2 * i + 1 ] = wagi [ i ];
     }
 srednia [ 2 * n - 2 ] = tab [ n - 1 ];
 cout << SredniaWazona ( srednia, n, c ) << " ";
 cout << c << endl;
 srednia [ 2 * n - 3 ] = 0.45;
 cout << SredniaWazona ( srednia, n, c ) << " ";
 cout << c << endl;

 double x1 = 1, x2 = 1;
 cout << PierwiastkiKwadratowe ( 1, 2, 1, x1, x2 );
 cout << " " << x1 << " " << x2 << endl;

 Fibonacci ( tab, 4 );
 for ( int i = 0; i <= 4; ++i )
     cout << tab [ i ] << " ";
 cout << endl;

 cout << Licznik ( true, n ) << " ";
 cout << n << endl;
 cout << Licznik ( true, n ) << " ";
 cout << n << endl;
 cout << Licznik ( false, n ) << " ";
 cout << n << endl;

 return 0;
}
```

Odpowiadającym wyjściem jest:

```
-3
1
1 0.75
0 0.75
1 -1 1
0 1 1 2 3
0 1
0 2
1 2
```