AiSD Projekt 1

Wygenerowano przez Doxygen 1.9.5

1 Zadanie 10.

1.0.0.1 Sprawdź, które elementy tablicy dwuwymiariowej występują w każdym wierszu tej tablicy.

1.0.1 Przykład.

Wejście:
[2,4,3,8,7]
[4,7,1,3,6]
[3,5,2,1,3]
[4,5,0,2,3]

Wyjście: 3

2 Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

3 Dokumentacja plików

3.1 Dokumentacja pliku projekt1.h

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <algorithm>
#include <time.h>
#include <vector>
#include <chrono>
#include <deque>
```

Definicje

- #define **PRO_FILE_VALUE_DELIMITER** ','
- #define **PRO_FILE_ARRAY_DELIMITER** '\n'

Funkcje

- void **pro::init** ()

  Inicjalizuje bibliotekę pomocniczą.
- int **pro::losowa_liczba** (int min, int max)

  Generuje losową liczbę z przedziału [min, max].
- std::vector< int > **pro::generuj_losowy_ciag** (int min, int max, int width)

  Generuje losowy ciag o podanej długości z wartościami z podanego przedziału.
- std::vector< std::vector< int > > **pro::generuj_losowy_ciag_2d** (int min, int max, int width, int height)

  Generuje losowy dwuwymiarowy ciag o podanych wymiarach z wartościami z podanego przedziału.
- std::vector< int > **pro::generuj_ciag_z_zakresu** (int start, int end, int step=1)

  Zwraca ciag z zakresu start do end z krokiem step.
- std::vector< int >::iterator **pro::quicksort_iterator_partition** (std::vector< int >::iterator begin, std::vector< int >::iterator end)

  Funkcja pomocnicza sortowania quicksort.
- void **pro::quicksort_iterator** (std::vector< int >::iterator begin, std::vector< int >::iterator end)

  Sortowanie metodą quicksort na podanym przedziale.
- std::vector< int >::iterator **pro::linear_search_iterator** (std::vector< int > &arr, int val)

  Przeprowadza wyszukiwanie liniowe w wartościach tablicy.
- std::vector< int >::iterator **pro::set_intersection** (const std::vector< int > &arr1, const std::vector< int > &arr2, std::vector< int >::iterator res)

  Wyszukuje wspólne elementy dwóch tablic.
- void **pro::opisz_ciag** (const std::vector< int > &arr)

  Wypisuje w konsoli wymiary tablicy.
- void **pro::opisz_ciag** (const std::vector< std::vector< int > > &arr)

  Wypisuje w konsoli wymiary tablicy.
- std::vector< int > **pro::odczytaj_ciag_z_pliku** (const char *nazwa_pliku, char delimiter=PRO_FILE_VALUE_DELIMITER)

  Odczytuje ciąg z pliku wejściowego.
- std::vector< std::vector< int > > **pro::odczytaj_ciag_2d_z_pliku** (const char *nazwa_pliku, char delimiter_val=PRO_FILE_VALUE_DELIMITER, char delimiter_array=PRO_FILE_ARRAY_DELIMITER)

  Odczytuje dwuwymiarową tablicę z pliku wejściowego.
- std::pair< std::vector< std::vector< int > >::const_iterator, std::vector< std::vector< int > >::const_iterator > **pro::thread_bounds** (const std::vector< std::vector< int > > &data, int thread_count, int thread_id)

  Oblicza zakres danych, na których mają być wykonane operacje dla podanego wątku.
- template<class T >
  void **pro::wypisz_ciag** (const std::vector< T > &arr, unsigned spacing=0)

  Wypisuje zawartość tablicy na ekranie.
- template<class T >
  void **pro::wypisz_ciag** (const std::vector< std::vector< T > > &data, unsigned spacing=0)

  Wypisuje zawartość tablicy dwuwymiarowej na ekranie.
- template<class T >
  void **pro::zapisz_ciag_do_pliku** (const char *nazwa_pliku, const std::vector< T > &data, char delimiter=PRO_FILE_VALUE_DELIMITER)

  Zapisuje ciąg do pliku wyjściowego.
- template<class T >
  void **pro::zapisz_ciag_2d_do_pliku** (const char *nazwa_pliku, const std::vector< std::vector< T > > &data, char delimiter_val=PRO_FILE_VALUE_DELIMITER, char delimiter_array=PRO_FILE_ARRAY_DELIMITER)

  Zapisuje tablilcę dwuwymiarową do pliku wyjściowego.

3.1.1 Opis szczegĂłĹ‚owy

Autor

Dariusz Strojny

Data

November 2022

3.1.2 Dokumentacja funkcji

3.1.2.1 generuj_ciag_z_zakresu()

```
std::vector< int > pro::generuj_ciag_z_zakresu (
        int start,
        int end,
        int step = 1 )
```

Zwraca ciag z zakresu start do end z krokiem step.
np. f(2, 6, 2) -> [2, 4, 6]

Parametry

start	Początkowa wartość iteratora
end	Maksymalna wartość iteratora (włącznie)
step	Krok o jaki zwiększany jest iterator

3.1.2.2 generuj_losowy_ciag()

```
std::vector< int > pro::generuj_losowy_ciag (
        int min,
        int max,
        int width )
```

Generuje losowy ciag o podanej długości z wartościami z podanego przedziału.

Parametry

min	Minimalna wartość elementu w ciągu
max	Maksymalna wartosć elementu w ciągu
width	Ilość elementów w ciągu

Zwraca

wygenerowany ciąg

3.1.2.3 generuj_losowy_ciag_2d()

```
std::vector< std::vector< int > > pro::generuj_losowy_ciag_2d (
            int min,
            int max,
            int width,
            int height )
```

Generuje losowy dwuwymiarowy ciag o podanych wymiarach z wartościami z podanego przedziału.

Parametry

min	Minimalna wartość elementu w ciągu
max	Maksymalna wartosć elementu w ciągu
width	Ilość kolumn w ciągu
height	Ilość wierszy ciągu

Zwraca

wygenerowany ciąg

3.1.2.4 linear_search_iterator()

```
std::vector< int >::iterator pro::linear_search_iterator (
            std::vector< int > & arr,
            int val )
```

Przeprowadza wyszukiwanie liniowe w wartościach tablicy.

Parametry

arr	Tablica, na której wykonywane jest wyszukiwanie
val	Wartość szukana w tablicy

Zwraca

Iterator wskazujący na znaleziony element lub na koniec przedziału

3.1.2.5 losowa_liczba()

```
int pro::losowa_liczba (
            int min,
            int max )  [inline]
```

Generuje losową liczbę z przedziału [min, max].

Parametry

min	Minimalna wartość liczby
max	Maksymalna wartosć liczby

Zwraca

wygenerowana liczba

3.1.2.6 odczytaj_ciag_2d_z_pliku()

```
std::vector< std::vector< int > > pro::odczytaj_ciag_2d_z_pliku (
        const char * nazwa_pliku,
        char delimiter_val = PRO_FILE_VALUE_DELIMITER,
        char delimiter_array = PRO_FILE_ARRAY_DELIMITER )
```

Odczytuje dwuwymiarową tablicę z pliku wejściowego.

Parametry

nazwa_pliku	Ścieżka do pliku
delimiter_val	Znak oddzielający wartości wiersza w pliku
delimiter_array	Znak oddzielający kolumny w pliku

Zwraca

Dwuwymiarowa tablica odczytany z pliku

3.1.2.7 odczytaj_ciag_z_pliku()

```
std::vector< int > pro::odczytaj_ciag_z_pliku (
        const char * nazwa_pliku,
        char delimiter = PRO_FILE_VALUE_DELIMITER )
```

Odczytuje ciąg z pliku wejściowego.

Parametry

nazwa_pliku	Ścieżka do pliku
delimiter	Znak oddzielający wartości w pliku

Zwraca

Ciąg odczytany z pliku

3.1.2.8 opisz_ciag() [1/2]

```
void pro::opisz_ciag (
        const std::vector< int > & arr )
```

Wypisuje w konsoli wymiary tablicy.

Parametry

arr	opisywana tablica

3.1.2.9 opisz_ciag() [2/2]

```
void pro::opisz_ciag (
        const std::vector< std::vector< int > > & arr )
```

Wypisuje w konsoli wymiary tablicy.

Parametry

arr	opisywana tablica

3.1.2.10 quicksort_iterator()

```
void pro::quicksort_iterator (
        std::vector< int >::iterator begin,
        std::vector< int >::iterator end )
```

Sortowanie metodą quicksort na podanym przedziale.

Parametry

begin	Iterator wskazujący na początek przedziału
end	Iterator wskazujący na koniec przedziału

3.1.2.11 quicksort_iterator_partition()

```
std::vector< int >::iterator pro::quicksort_iterator_↩
partition (
        std::vector< int >::iterator begin,
        std::vector< int >::iterator end )
```

Funkcja pomocnicza sortowania quicksort.

Dzieli ciąg danych na dwie części przenosząc elementy mniejsze lub równe pierwszej wartości na jej lewą stronę a pozostałe na jej prawą stronę.

Parametry

begin	
end	

Zwraca

Iterator wskazujący na wartość oddzielającą oba ciągi

3.1.2.12 set_intersection()

```
std::vector< int >::iterator pro::set_intersection (
        const std::vector< int > & arr1,
        const std::vector< int > & arr2,
        std::vector< int >::iterator res )
```

Wyszukuje wspólne elementy dwóch tablic.

Funkcja wykonuje wyszukiwanie wspólnych elementów poprzez skrzyżowanie ze sobą dwóch tablic. Tablice wejściowe muszą być posortowane rosnąco.

Parametry

arr1	Pierwsza tablica
arr2	Druga tablica
res	Iterator wskazujący na pierwszy element tablicy o rozmiarze przynajmniej min(rozmiar arr1, rozmiar arr2)

Zwraca

Iterator wskazujący na element za ostatnim wpisanym elementem

3.1.2.13 thread_bounds()

```
std::pair< std::vector< std::vector< int > >::const_iterator, std::vector< std::vector< int > >::const_iterator > pro::thread_bounds (
        const std::vector< std::vector< int > > & data,
        int thread_count,
        int thread_id )
```

Oblicza zakres danych, na których mają być wykonane operacje dla podanego wątku.

Parametry

data	Dane do podzielenia
thread_count	Łączna ilość wątków
thread_id	Numer wątku, dla którego obliczany jest zakres

Zwraca

Para iteratorów wskazujących na początek i koniec wyznaczonego zakresu danych

3.1.2.14 wypisz_ciag() [1/2]

```
template<class T >
void pro::wypisz_ciag (
        const std::vector< std::vector< T > > & data,
        unsigned spacing = 0 )
```

Wypisuje zawartość tablicy dwuwymiarowej na ekranie.

Parametry Szablonu

T	Rodzaj danych przechowywanych w tablicy

Parametry

data	Tablica do wyświetlenia
spacing	Dopełnienie każdej komórki danych znakami białymi do podanej ilości znaków

3.1.2.15 wypisz_ciag() [2/2] template<class T >

```
void pro::wypisz_ciag (
            const std::vector< T > & arr,
            unsigned spacing = 0 )
```

Wypisuje zawartość tablicy na ekranie.

Parametry Szablonu

T	Rodzaj danych przechowywanych w tablicy

Parametry

arr	Tablica do wyświetlenia
spacing	Dopełnienie każdej komórki danych znakami białymi do podanej ilości znaków

3.1.2.16 zapisz_ciag_2d_do_pliku() template<class T >

```
void pro::zapisz_ciag_2d_do_pliku (
            const char * nazwa_pliku,
            const std::vector< std::vector< T > > & data,
            char delimiter_val = PRO_FILE_VALUE_DELIMITER,
            char delimiter_array = PRO_FILE_ARRAY_DELIMITER )
```

Zapisuje tablilcę dwuwymiarową do pliku wyjściowego.

Parametry Szablonu

T	Rodzaj danych przechowywanych w tablicy

Parametry

nazwa_pliku	Ścieżka do pliku
data	Tablica do zapisania
delimiter_val	Znak oddzielający wartości wiersza w pliku
delimiter_array	Znak oddzielający kolumny w pliku

3.1.2.17 zapisz_ciag_do_pliku() template<class T >

```
void pro::zapisz_ciag_do_pliku (
            const char * nazwa_pliku,
            const std::vector< T > & data,
            char delimiter = PRO_FILE_VALUE_DELIMITER )
```

Zapisuje ciąg do pliku wyjściowego.

Parametry Szablonu

T	Rodzaj danych przechowywanych w ciągu

Parametry

nazwa_pliku	Ścieżka do pliku
data	Ciąg do zapisania
delimiter	Znak oddzielający wartości w pliku

3.2 projekt1.h

IdĹş do dokumentacji tego pliku.

```
1
9 #ifndef __PROJEKT_1_AISD__
10 #define __PROJEKT_1_AISD__
11
12 #include <iostream>
13 #include <fstream>
14 #include <sstream>
15 #include <string>
16
17 #include <algorithm>
18
19 #include <time.h>
20
21 #include <vector>
22 #include <chrono>
23
24 #include <deque>
25
26 #define PRO_FILE_VALUE_DELIMITER ','
27 #define PRO_FILE_ARRAY_DELIMITER '\n'
28
29 namespace pro
30 {
31     /* --------   FUNCTION DECLARATIONS   ---------- */
32
36     void init();
37
46     inline int losowa_liczba(int min, int max);
47
48     //
58     std::vector<int> generuj_losowy_ciag(int min, int max, int width);
59
70     std::vector<std::vector<int» generuj_losowy_ciag_2d(int min, int max, int width, int height);
71
83     std::vector<int> generuj_ciag_z_zakresu(int start, int end, int step = 1);
84
95     std::vector<int>::iterator quicksort_iterator_partition(std::vector<int>::iterator begin,
       std::vector<int>::iterator end);
96
103     void quicksort_iterator(std::vector<int>::iterator begin, std::vector<int>::iterator end);
104
113     std::vector<int>::iterator linear_search_iterator(std::vector<int>& arr, int val);
114
```

```
115   // wyszukuje wspólne elementy tablic arr1 i arr2 poprzez intersekcje oraz przepisuje je do tablicy
      res
116   // zwraca iterator tablicy res wskazujący na ostatni przypisany element
117
130   std::vector<int>::iterator set_intersection(const std::vector<int>& arr1, const std::vector<int>&
      arr2, std::vector<int>::iterator res);
131
137   void opisz_ciag(const std::vector<int>& arr);
138
144   void opisz_ciag(const std::vector<std::vector<int»& arr);
145
154   std::vector<int> odczytaj_ciag_z_pliku(const char* nazwa_pliku, char delimiter =
      PRO_FILE_VALUE_DELIMITER);
155
165   std::vector<std::vector<int» odczytaj_ciag_2d_z_pliku(const char* nazwa_pliku, char delimiter_val =
      PRO_FILE_VALUE_DELIMITER, char delimiter_array = PRO_FILE_ARRAY_DELIMITER);
166
176   std::pair<std::vector<std::vector<int»::const_iterator,
      std::vector<std::vector<int»::const_iterator>
177       thread_bounds(const std::vector<std::vector<int»& data, int thread_count, int thread_id);
178
179
180   /* ---======-   TEMPLATE FUNCTION DECLARATIONS   -=======- */
181
189   template<class T>
190   void wypisz_ciag(const std::vector<T>& arr, unsigned spacing = 0);
191
199   template<class T>
200   void wypisz_ciag(const std::vector<std::vector<T»& data, unsigned spacing = 0);
201
210   template<class T>
211   void zapisz_ciag_do_pliku(const char* nazwa_pliku, const std::vector<T>& data, char delimiter =
      PRO_FILE_VALUE_DELIMITER);
212
222   template<class T>
223   void zapisz_ciag_2d_do_pliku(const char* nazwa_pliku, const std::vector<std::vector<T»& data, char
      delimiter_val = PRO_FILE_VALUE_DELIMITER, char delimiter_array = PRO_FILE_ARRAY_DELIMITER);
224
225
226   /* ---======-   TEMPLATE FUNCTION DEFINITIONS   -=======- */
227
228   // wypisuje tablice na ekranie
229   template<class T>
230   void wypisz_ciag(const std::vector<T>& arr, unsigned spacing)
231   {
232       std::cout « "[";
233
234       // jeżeli przekazana została domyślna długość dopełnienia
235       if (spacing == 0)
236       {
237           // dla każdego elementu tablicy
238           for (auto el = arr.begin(); el != arr.end(); el++)
239           {
240               // wypisanie wartości elementu
241               std::cout « *el;
242               // dla wartości innych niż ostatnia wypisz znak ','
243               if (el != arr.end() - 1) std::cout « ",";
244           }
245       }
246       // w przeciwnym wypadku
247       else
248       {
249           // zabezpieczenie przed przypadkowym przepełnieniem w dół (unsigned -1 = 4294967295)
250           if (spacing > 50) spacing = 50;
251
252
253           // utworzenie tablicy znaków dla formatu dopełnienia wartości
254           char* mod = new char[12];
255           // wpisanie formatu do tablicy znaków (np. "%3d, ")
256           sprintf_s(mod, 12, "%%%dd", spacing);
257
258           // dla każdego elementu tablicy
259           for (auto el = arr.begin(); el != arr.end(); el++)
260           {
261               // wypisanie wartości elementu przy użyciu utworzonego wcześniej formatu
262               printf(mod, *el);
263               if (el != arr.end() - 1) std::cout « ",";
264           }
265
266           // zwolnienie pamięci tablicy formatu
267           delete[] mod;
268       }
269
270       std::cout « "]\n";
271   }
272
273   template<class T>
```

```
274     void wypisz_ciag(const std::vector<std::vector<T>& data, unsigned spacing)
275     {
276         // dla każdego elementu tablicy 2-wymiarowej
277         for (auto const& arr : data)
278         {
279             // wypisz wartości ciągu 1-wymiarowego wykorzystując istniającą funkcję wypisz_ciąg
280             pro::wypisz_ciag(arr, spacing);
281         }
282     }
283 
284     // zapisuje tablice do pliku wyjsciowego z opcjonalną specyfikacją znaku oddzielającego warości
285     template<class T>
286     void zapisz_ciag_do_pliku(const char* nazwa_pliku, const std::vector<T>& data, char delimiter)
287     {
288         // otwarcie pliku do zapisu
289         std::fstream ofs(nazwa_pliku, std::ios::out);
290 
291         // weryfikacja otwarcia pliku
292         if (!ofs.good())
293             // błąd przy próbie otwarcia pliku
294             throw std::string("Nie udalo sie otworzyc pliku ") + nazwa_pliku + " do zapisu!";
295 
296         // dla każdego elementu tablicy
297         for (const auto& el : data)
298             // wpisane wartości do pliku razem ze znakiem końca wartości
299             ofs « el « delimiter;
300     }
301 
302     // zapisuje tablice dwuwymiarową do pliku wyjsciowego z opcjonalną specyfikacją znaku oddzielającego
    warości i tablice
303     template<class T>
304     void zapisz_ciag_2d_do_pliku(const char* nazwa_pliku, const std::vector<std::vector<T>& data, char
    delimiter_val, char delimiter_array)
305     {
306         // otwarcie pliku do zapisu
307         std::fstream ofs(nazwa_pliku, std::ios::out);
308 
309         // weryfikacja otwarcia pliku
310         if (!ofs.good())
311             // błąd przy próbie otwarcia pliku
312             throw std::string("Nie udalo sie otworzyc pliku ") + nazwa_pliku + " do zapisu!";
313 
314         // dla każdego podciągu
315         for (const auto& arr : data)
316         {
317             //dla każdego elementu tablicy
318             for (const auto& el : arr)
319             {
320                 // wpisane wartości do pliku razem ze znakiem końca wartości
321                 ofs « el « delimiter_val;
322             }
323 
324             // wpisane znaku końca tabeli
325             ofs « delimiter_array;
326         }
327 
328     }
329 }
330 
331 #endif
```

Skorowidz