AiSD Projekt 1

Wygenerowano przez Doxygen 1.9.6

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

  Zwraca ciąg 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::pair< std::vector< int >::iterator, std::vector< int >::iterator > **pro::quicksort_iterator_three_way_partition** (std::vector< int >::iterator start, std::vector< int >::iterator end)

  Funkcja pomocnicza sortowania quicksort wykorzystująca usprawnienie dla ciągów z często powtarzającymi się wartościami.
- void **pro::quicksort_three_way_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 tablicę dwuwymiarową do pliku wyjściowego.

3.1.1 Opis szczegółowy

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 ciąg 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 ciąg o podanej długości z wartościami z podanego przedziału.

Parametry

min	Minimalna wartość elementu w ciągu

Parametry

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 ciąg 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 wiersze w pliku

Zwraca

Dwuwymiarowa tablica odczytana 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 quicksort_iterator_three_way_partition()

```
std::pair< std::vector< int >::iterator, std::vector< int >::iterator > pro::quicksort_iterator_three_way_partition (
        std::vector< int >::iterator start,
        std::vector< int >::iterator end )
```

Funkcja pomocnicza sortowania quicksort wykorzystująca usprawnienie dla ciągów z często powtarzającymi się wartościami.

Dzieli ciąg danych na trzy części przenosząc elementy mniejsze pierwszej wartości na jej lewą stronę, większe od niej na jej prawą stronę a równe jej na środek.

Parametry

begin	
end	

Zwraca

Para iteratorów wskazujących odpowiednio na koniec i początek przedziałów oddzielonych ciągiem złożonym z wartości równych wybranej wartości pivot.

3.1.2.13 quicksort_three_way_iterator()

```
void pro::quicksort_three_way_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

3.1.2.14 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

3.1.2.15 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	£ączna ilość wątków
thread_id	Numer wątku, dla którego obliczany jest zakres

Zwraca

Para iteratorów wskazujących na początek i koniec wyznaczonego zakresu danych

3.1.2.16 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

3.1.2.17 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

3.1.2.18 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 tablicę 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 wiersze w pliku

3.1.2.19 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

Idź do dokumentacji tego pliku.

```
00001
00009 #ifndef __PROJEKT_1_AISD__
00010 #define __PROJEKT_1_AISD__
00011
00012 #include <Windows.h>
00013
00014 #include <iostream>
00015 #include <fstream>
00016 #include <sstream>
00017 #include <string>
00018
00019 #include <algorithm>
00020
00021 #include <time.h>
00022
00023 #include <vector>
00024 #include <chrono>
00025
00026 #include <deque>
00027
00028 #define PRO_FILE_VALUE_DELIMITER ' '
00029 #define PRO_FILE_ARRAY_DELIMITER '\n'
00030
00031 namespace pro
00032 {
00033     /* ---===---   FUNCTION DECLARATIONS   ---===--- */
00034
00038     void init();
00039
00048     inline int losowa_liczba(int min, int max);
00049
00050     //
00060     std::vector<int> generuj_losowy_ciag(int min, int max, int width);
00061
00072     std::vector<std::vector<int» generuj_losowy_ciag_2d(int min, int max, int width, int height);
00073
00085     std::vector<int> generuj_ciag_z_zakresu(int start, int end, int step = 1);
00086
00097     std::vector<int>::iterator quicksort_iterator_partition(std::vector<int>::iterator begin,
      std::vector<int>::iterator end);
00098
00105     void quicksort_iterator(std::vector<int>::iterator begin, std::vector<int>::iterator end);
00106
00117     std::pair<std::vector<int>::iterator, std::vector<int>::iterator>
      quicksort_iterator_three_way_partition(std::vector<int>::iterator start, std::vector<int>::iterator
      end);
00118
00125     void quicksort_three_way_iterator(std::vector<int>::iterator begin, std::vector<int>::iterator
      end);
```

```
00134     std::vector<int>::iterator linear_search_iterator(std::vector<int>& arr, int val);
00135
00136     // wyszukuje wspólne elementy tablic arr1 i arr2 poprzez intersekcje oraz przepisuje je do tablicy
      res
00137     // zwraca iterator tablicy res wskazujący na ostatni przypisany element
00138
00151     std::vector<int>::iterator set_intersection(const std::vector<int>& arr1, const std::vector<int>&
      arr2, std::vector<int>::iterator res);
00152
00158     void opisz_ciag(const std::vector<int>& arr);
00159
00165     void opisz_ciag(const std::vector<std::vector<int>& arr);
00166
00175     std::vector<int> odczytaj_ciag_z_pliku(const char* nazwa_pliku, char delimiter =
      PRO_FILE_VALUE_DELIMITER);
00176
00186     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);
00187
00197     std::pair<std::vector<std::vector<int>::const_iterator,
      std::vector<std::vector<int>::const_iterator>
00198         thread_bounds(const std::vector<std::vector<int>& data, int thread_count, int thread_id);
00199
00200
00201     /* --------   TEMPLATE FUNCTION DECLARATIONS   -------- */
00202
00210     template<class T>
00211     void wypisz_ciag(const std::vector<T>& arr, unsigned spacing = 0);
00212
00220     template<class T>
00221     void wypisz_ciag(const std::vector<std::vector<T>& data, unsigned spacing = 0);
00222
00231     template<class T>
00232     void zapisz_ciag_do_pliku(const char* nazwa_pliku, const std::vector<T>& data, char delimiter =
      PRO_FILE_VALUE_DELIMITER);
00233
00243     template<class T>
00244     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);
00245
00246
00247     /* --------   TEMPLATE FUNCTION DEFINITIONS   -------- */
00248
00249     // wypisuje tablice na ekranie
00250     template<class T>
00251     void wypisz_ciag(const std::vector<T>& arr, unsigned spacing)
00252     {
00253         std::cout << "[";
00254
00255         // jeżeli przekazana została domyślna długość dopełnienia
00256         if (spacing == 0)
00257         {
00258             // dla każdego elementu tablicy
00259             for (auto el = arr.begin(); el != arr.end(); el++)
00260             {
00261                 // wypisanie wartości elementu
00262                 std::cout << *el;
00263                 // dla wartości innych niż ostatnia wypisz znak ','
00264                 if (el != arr.end() - 1) std::cout << ",";
00265             }
00266         }
00267         // w przeciwnym wypadku
00268         else
00269         {
00270             // zabezpieczenie przed przypadkowym przepełnieniem w dół (unsigned -1 = 4294967295)
00271             if (spacing > 50) spacing = 50;
00272
00273
00274             // utworzenie tablicy znaków dla formatu dopełnienia wartości
00275             char* mod = new char[12];
00276             // wpisanie formatu do tablicy znaków (np. "%3d, ")
00277             sprintf_s(mod, 12, "%%%dd", spacing);
00278
00279             // dla każdego elementu tablicy
00280             for (auto el = arr.begin(); el != arr.end(); el++)
00281             {
00282                 // wypisanie wartości elementu przy użyciu utworzonego wcześniej formatu
00283                 printf(mod, *el);
00284                 if (el != arr.end() - 1) std::cout << ",";
00285             }
00286
00287             // zwolnienie pamięci tablicy formatu
00288             delete[] mod;
00289         }
00290
00291         std::cout << "]\n";
00292     }
```

```
00293
00294     template<class T>
00295     void wypisz_ciag(const std::vector<std::vector<T»& data, unsigned spacing)
00296     {
00297         // dla każdego elementu tablicy 2-wymiarowej
00298         for (auto const& arr : data)
00299         {
00300             // wypisz wartości ciągu 1-wymiarowego wykorzystując istniającą funkcję wypisz_ciąg
00301             pro::wypisz_ciag(arr, spacing);
00302         }
00303     }
00304
00305     // zapisuje tablice do pliku wyjsciowego z opcjonalną specyfikacją znaku oddzielającego warości
00306     template<class T>
00307     void zapisz_ciag_do_pliku(const char* nazwa_pliku, const std::vector<T>& data, char delimiter)
00308     {
00309         // otwarcie pliku do zapisu
00310         std::fstream ofs(nazwa_pliku, std::ios::out);
00311
00312         // weryfikacja otwarcia pliku
00313         if (!ofs.good())
00314             // błąd przy próbie otwarcia pliku
00315             throw std::string("Nie udalo sie otworzyc pliku ") + nazwa_pliku + " do zapisu!";
00316
00317         // dla każdego elementu tablicy
00318         for (const auto& el : data)
00319             // wpisane wartości do pliku razem ze znakiem końca wartości
00320             ofs « el « delimiter;
00321     }
00322
00323     // zapisuje tablice dwuwymiarową do pliku wyjsciowego z opcjonalną specyfikacją znaku
      oddzielającego warości i tablice
00324     template<class T>
00325     void zapisz_ciag_2d_do_pliku(const char* nazwa_pliku, const std::vector<std::vector<T»& data, char
      delimiter_val, char delimiter_array)
00326     {
00327         // otwarcie pliku do zapisu
00328         std::fstream ofs(nazwa_pliku, std::ios::out);
00329
00330         // weryfikacja otwarcia pliku
00331         if (!ofs.good())
00332         {
00333             // jeżeli wartość błędu jest równa 2, to ścieżka do pliku jest niepoprawna lub nie
      istnieje
00334             // tworzenie folderu wskazanego przez ścieżkę
00335             if (errno == 2)
00336             {
00337                 std::string fn = std::string(nazwa_pliku);
00338                 size_t pos = fn.rfind("/");
00339                 if (pos == std::string::npos)
00340                     pos = fn.rfind("\\");
00341
00342                 std::string cmd = std::string("mkdir ");
00343                 if (pos == std::string::npos) cmd += std::string(nazwa_pliku);
00344                 else cmd += std::string(nazwa_pliku).substr(0, pos);
00345
00346                 std::cout « "Executing \"" « cmd « "\"\n";
00347                 system(cmd.c_str());
00348
00349                 ofs.open(nazwa_pliku, std::ios::out);
00350             }
00351
00352             if(!ofs.good())
00353                 // błąd przy próbie otwarcia pliku
00354                 throw std::string("Nie udalo sie otworzyc pliku ") + nazwa_pliku + " do zapisu! Kod
      bledu: " + std::to_string(errno);
00355         }
00356
00357         // dla każdego podciągu
00358         for (const auto& arr : data)
00359         {
00360             //dla każdego elementu tablicy
00361             for (const auto& el : arr)
00362             {
00363                 // wpisane wartości do pliku razem ze znakiem końca wartości
00364                 ofs « el « delimiter_val;
00365             }
00366
00367             // wpisane znaku końca tabeli
00368             ofs « delimiter_array;
00369         }
00370
00371     }
00372 }
00373
00374 #endif
```

Skorowidz