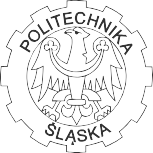
Politechnika Śląska w Gliwichach

Wydział Automatyki, Elektroniki i Informatyki

**Podstawy Programowania**

**Komputerów**

Temat projektu

**Maraton**

autor Szymon Myszor

prowadzący dr inż. Imię Nazwisko

rok akademicki 2018/2019

kierunek Teleinformatyka

rodzaj studiów SSI

semestr 1

termin laboratorium / ćwiczeń piątek, 08:30 – 10:00 grupa

sekcja 7

termin oddania sprawozdania

data oddania sprawozdania

# Treść zadania

Napisać program zwracający pliki tekstowe, z zawartymi w nich danymi maratończyka oraz posortowanymi według daty maratonów, w których uczestniczył. Do tego celu wykorzystać listę list. Dane do wypisania zawarte są w plikach maratonów zawięracych nazwę i datę maratonu i listę maratończyków, którzy brali w nim udział. Dane każdego maratończyka znajdują się w osobnej linii. Program uruchamiany jeset z linii poleceń z wykorzystaniem przełącznika:

-i pliki wejściowe z protokołami zawodów

# Analiza zadania

Zagadnienie przedstawia problem budowy diagramu w postaci listy list i posortowania danych w podliście każdego elementu listy. Dane do budowy diagramu odczytywane są z plików tekstowych.

## Struktury danych

W programie wykorzystano listę list do przechowywania danych. Główna lista przechowuje dane maratończyka, natomiast każda podlista danego maratończyka zawiera dane o maratonach w jakich uczestniczył wraz z uzyskanymi czasami. Podlista jest posortowana rosnąco według dat maratonów. Taka struktura danych umożliwia łatwy dostęp do wszystkich danych poszczególnego maratończyka. Rysunek 1 przedstawia przykład listy list.

## Algorytmy

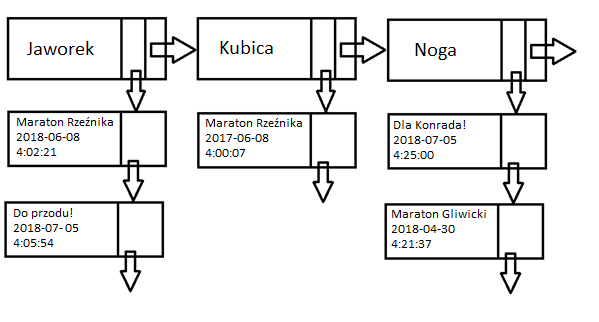
Program sortuje daty maratonów umieszczonych w podlistach maratończyków. Wstawianie kolejnych maratonów jest realizowane poprzez rekurencyjne przejście przez podlistę. Utworzenie podlisty i przejście przez nią jest wykonywane w średnim czasie *O(n*2*)*.

# Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Należy przekazać do programu nazwy plików wejściowych po odpowiednim przełączniku (-i). Np:

Maraton.exe –i Maraton1.txt Maraton2.txt Maraton3.txt Maraton4.txt

Rysunek 2: Przykład listy list, która może przechowywać dane pobrane z plików. W programie główna lista przechowuje dane o maratończykach, natomiast podlisty przechowują dane o maratonach, w których dany maratończyk brał udział.



Rysunek 1.

Pliki są plikami tekstowymi, ale mogą mieć dowolne rozszerzenie (lub go nie mieć). Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie kominkatu o błędzie.

# Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (budowy grafu i wykonywania na nim operacji).

## Typy zdefiniowane w programie

W programie zdefiniowano następujące typy:

struct maratonczyk

{

string nazwisko;

maraton \*pMaraton; //wskaznik na podliste

maratonczyk \*pNext;

};

Typ ten służy do budowy głównej listy – listy maratończyków.

struct maraton

{

string nazwa;

string data;

string czas;

maraton \*pNext;

};

Ten typ służy do budowy podlisty maratonów dla każdego elementu głównej listy maratończyków.

## Ogólna struktura program

W funkcji głównej (main) wywołane są następujące funkcje:

bool sprawdzanie\_argumentow(int argc, char \*\*argv, string plikiMaratonowNazwy[]);

Ta funkcja sprawdza, czy program został wywołany w prawidłowy sposób.

Następna funkcja zdefiniowana w programie to:

bool czytajDane(string nazwaPliku, maratonczyk \*& pMaratonczyk);

Funckja odczytuje dane z plików maratonów i przekazuje dalej, w celu budowy grafu.

Kolejna metoda zdefiniowana w programie to:

void budujGraf(maratonczyk \*& pMaratonczyk, const string &nazwisko,

const string &data, const string &nazwaMaratonu, const string &czas);

Funkcja buduje graf – listę list za pomocą metod opisanych poniżej.

maratonczyk \* znajdzLubDodajMaratonczyka(maratonczyk \*&pMaratonczyk,

const string &nazwisko);

Funkcja znajduje maratończyka w liście maratończyków lub dodaje go do listy.

void dodajMaraton(maraton \*&pMaraton, const string &nazwaMaratonu, const string &data, const string &czas);

Metoda dodaje maraton do listy maratonów maratończyka.

Kolejną funkcją w programie jest:

bool wypiszDoPliku(maratonczyk \*&pMaratonczyk);

Funkcja wypisuje dane do poszczególnych plików dane z grafu.

Ostatnią funkcją programu jest funkcja:

void usun(maratonczyk \*&pMaratonczyk);

Metoda usuwa graf.

## Szczegółowy opis implementacji funkcji

bool sprawdzanie\_argumentow(int argc, char \*\*argv, string plikiMaratonowNazwy[]);

Ta funkcja sprawdza, czy program został wywołany w prawidłowy sposób. Przyjmuje jako parametry libczę argumentów podanych podczas wywoływania programu i dynamiczną tablicę stringów, do której zapisywane są nazwy plików maratonów podawanych w parametrach uruchomieniowych. Funkcja zwraca wartość bool. True – dla poprawnie wpisanych parametrów, False – dla niepoprawnie wpisanych parametrów.

bool czytajDane(string nazwaPliku, maratonczyk \*& pMaratonczyk);

Funckja jako parametry przyjmuje nazwę pliku, który należy otworzyć oraz głowę listy. Metoda odczytuje dane z aktualnie otwartego pliku maratonu. Odczytane dane przekazuje do kolejnej metody (opisanej poniżej). Funkcja zwraca wartość bool. True – jeśli odczytach danych z pliku się powiódł, False – jeśli odczyt danych się nie powiódł.

void budujGraf(maratonczyk \*& pMaratonczyk, const string &nazwisko,

const string &data, const string &nazwaMaratonu, const string &czas);

Funkcja jako parametry przyjmuje dane odczytane z pliku (nazwisko maratończyka, czas maratończyka, data maratonu, nazwę maratonu) i głowę listy maratończyków. Metoda dodaje pierwszy element listy maratończyków wraz z danymi o pierwszych zawodach maratończyka lub (jeśli pierwszy element już istnieje) wywołuje kolejne funkcje dopowiedzialne za budowę listy list, które opisane są poniżej.

maratonczyk \* znajdzLubDodajMaratonczyka(maratonczyk \*&pMaratonczyk,

const string &nazwisko);

Funkcja przyjmuje jako parametry dane maratończyka oraz głowę listy zawodników i znajduje go w grafie, bądź w przypadku, gdy dany maratończyk pojawia się pierwszy raz dodaje go do listy zawodników. Metoda zwraca wskaźnik na znalezionego, bądź dodanego maratończyka, by później można było dodać do jego podlisty maraton, w którym uczestniczył.

void dodajMaraton(maraton \*&pMaraton, const string &nazwaMaratonu, const string &data, const string &czas);

Metoda jako parametry przyjmuje pierwszy element podlisty maratonów znalezionego/dodanego za pomocą funkcji znajdzLubDodajMaratonczyka i dodaje do jego podlisty zawodów informacje o maratonie odczytanym z pliku maratonu.

bool wypiszDoPliku(maratończyk \*&pMaratonczyk);

Funkcja przyjmuje jako parametr element listy maratończyków i wypisuje do poszczególnych plików (zgodnie z treścią zadania) informacje dotyczące wyników z zawodów, w których brał udział każdy znajdujący się w liście biegacz. Zwracaną przez metodę wartością jest bool. True – jeśli wypisanie danych do pliku się powiodło, False – jeśli wypisanie danych się nie powiodło.

void usun(maratończyk \*&pMaratonczyk);

Metoda przyjmuje jako parametr głowę grafu i iterując za pomocą dwóch pętli przechodzi po całym grafie i usuwa wszystkie jego elementy.

# Testowanie

Program został przetestowany na różnego rodzaju plikach tekstowych. Pusty plik a także zawarte dodatkowe białe znaki w pliku nie spowodują błędu programu. Wprowadzenie plików wejściowych bez odpowiednich rozszerzeń spowoduje wyświetlenie komunikatu o niepowodzeniu w otworzeniu plików. Wywołanie programu przy użyciu innego przełącznika niż „-i” spowoduje wyświetlenie komunikatu o błędnym podaniu przełącznika. Ilość maratończyków w jednym pliku nie ma znacznia.

# Wnioski

Największą trudność podczas pisania projektu sprawiło mi utworzenie funkcji, która doda maraton do podlisty maratończyka w odpowiedniej kolejności. Pisząc program zrozumiałem działanie list i nauczyłem się korzystania ze wskaźników.