**Dokumentacja projektu**

PRI - zadanie laboratoryjne 2

**Autor:**

Kamil Lipski

Warszawa, 2016

1. **Cele**

Program pozwala gromadzić w uporządkowanej tablicy liczb informacje o najlepszych „czasach” uzyskiwanych przez sprinterów podczas treningu w poszczególnych biegach. Do gromadzenia tych informacji wykorzystywana jest tablica o stałym rozmiarze 100 elementów. Program zapewnia pełną modyfikację (dodawanie, usuwanie, edytowanie) danych oraz umożliwia sortowanie po czasach z uwzględnieniem dystansu. Istnieje również możliwość sortowania po nazwiskach.

1. **Opis działania programu**

*Zmienne:* pozycja, dystans, czas - zmienne używane   
w strukturze (gromadzą dane zgodne z nazwami), zamiana, zamiana2 - zmienne używane przy zamianie elementów tablicy, numer - zmienna służąca do wprowadzania pozycji zawodnika, n - zmienna służąca do wprowadzania nowych elementów tablicy, jak również do badania poprzednich elementów, k, i - zmienne wykorzystywane przy edycji/usuwaniu/sprawdzaniu elementów tablicy, blad - zmienna służąca do sprawdzania poprawności wprowadzonych danych, min - zmienna wykorzystywana przy sortowaniu, petla1, petla2 - zmienne wykorzystywane przy pętlach, komenda - zmienna służąca do komunikacji użytkownika z programem

*Tablice:* nazwisko, imie - tablice przechowujące dane zgodne z nazwami, lista (tablica struktur) - dane zawodników, pomocnicza, usun\_edytuj (tablice struktur) - tablice wykorzystywane przy edycji/usuwaniu elementów tablicy

*Nowe typy:* zawodnik - struktura przechowująca dane zawodników, dane - struktura przechowująca dane podane przez użytkownika przy edycji/usuwaniu elementów tablicy

*Użyte funkcje:*

* stricmp(a, b) - funkcja służąca do porównywania ciągów znaków (bez uwzględniania wielkości liter) - zwraca *0*, gdy ciągi są identyczne, zwraca wartość większą od 0, gdy a>b i mniejszą od 0, gdy a<b
* sortuj\_dane(zawodnik a, zawodnik b) - służy do sortowania po nazwiskach. Zwraca *0*, jeśli a.nazwisko<b.nazwisko (lub a.imie<b.imie w przypadku identyczności nazwisk), a *1* w przeciwnym przypadku
* sortuj\_czasy(zawodnik a, zawodnik b) - służy do sortowania po czasach. Zwraca *0*, jeśli a.czas<b.czas, a *1*, gdy a.czas>b.czas. Jeśli czasy są takie same, wywołuje funkcję sortuj\_dane(a,b) i zwraca zwracaną przez wywoływaną funkcję wartość
* sortuj\_t(zawodnik a, zawodnik b) - służy do sortowania po czasach. Zwraca *0*, jeśli a.dystans<b.dystans, a *1*, gdy a.dystans>b.dystans. Jeśli dystanse są takie same, wywołuje funkcję sortuj\_czasy(a,b) i zwraca zwracaną przez wywoływaną funkcję wartość

Użytkownik podaje jedną z komend wymienionych w pro- gramie. Komenda *0* pozwala wprowadzić dane nowego zawodnika - prosi użytkownika o nazwisko, imię, dystans, i czas oraz przyporządkowuje zawodnikowi pozycję. Komenda *1*, pozwala posortować listę po nazwiskach - cała procedura jest wykonywana za pomocą algorytmu sortowania przez wybór. Po zakończeniu sortowania, program wypisuje posortowane dane. Komenda *2* działa   
na podobnej zasadzie, co poprzednia - sortuje tablicę, wykorzystując ten sam algorytm i wypisuje posortowane dane. Komenda *3* pozwala edytować wprowadzone dane - program prosi o podanie nazwiska i imienia biegacza, którego dane chcemy edytować, sprawdza, czy jest on w bazie i pozwala wprowadzić nowe dane. Komenda *4* usuwa dane zawodnika, którego nazwisko i imię podamy (jeśli znajduje się w bazie). Komenda *5* kończy działanie programu.

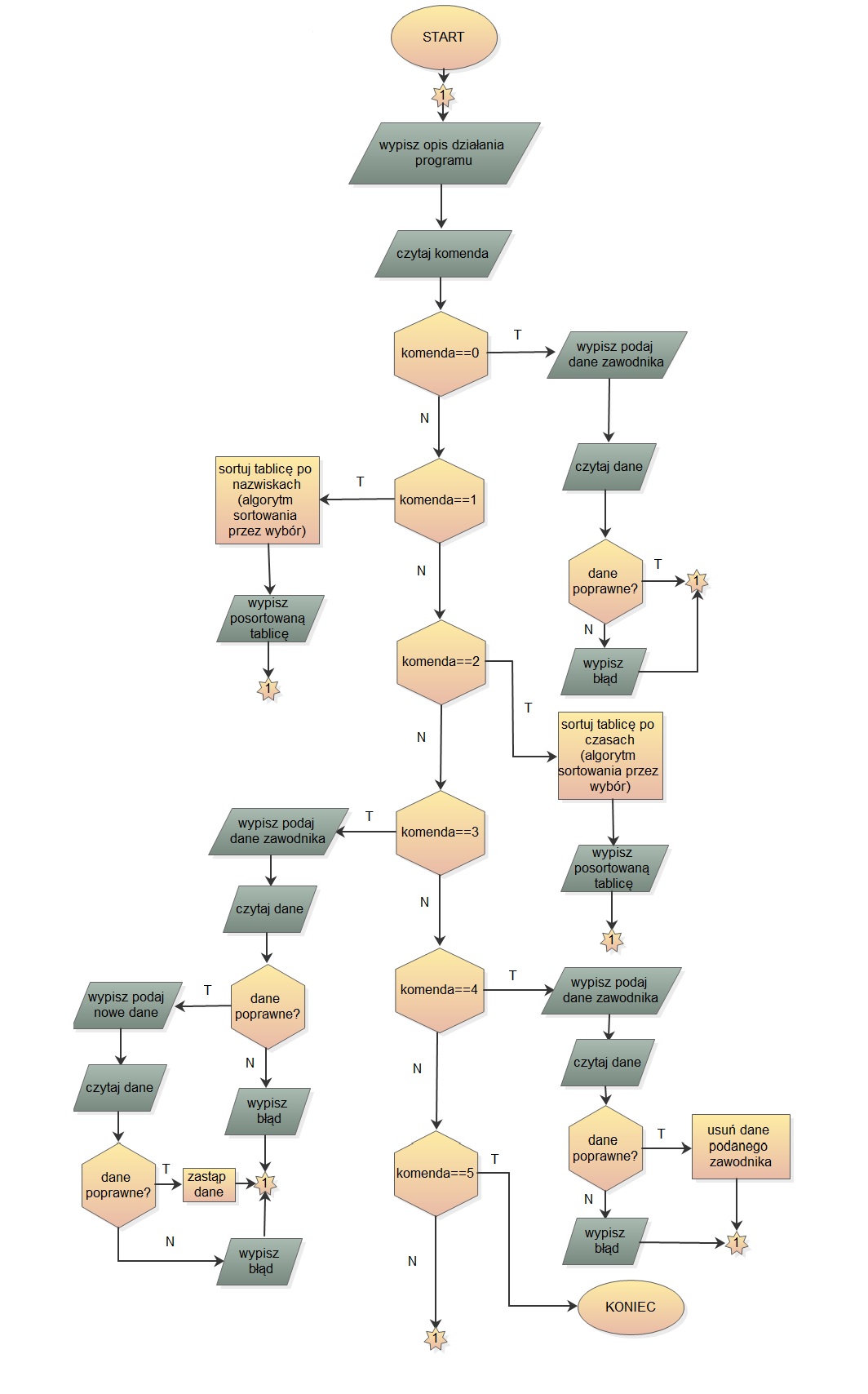
**3. Założenia**

* użytkownik podaje komendy wyświetlone przez program - w przeciwnym przypadku program nic nie wykona,
* nazwisko i imię rozpoczynają się wielką literą, kolejne są małe,
* zakładam, że długość nazwiska nie przekroczy 28 znaków, a długość imienia - 13; w przeciwnym przypadku program nie skopiuje całego ciągu znaków,
* program sprawdza poprawność wpisywanych danych, t.j.: dystans, czas. Gdy nie są poprawne, wyświetla komunikat o błędzie,
* program sprawdza, czy wprowadzane dane nie istnieją już w bazie. Jeśli tak, to zakomunikuje, że w bazie znajduje się już zawodnik o takich danych i zawodnik nie zostanie dodany,
* jeśli dane podane przy edycji/usuwaniu elementu tablicy nie zostaną znalezione w bazie, program wypisze komunikat: *Nie ma takiego zawodnika!,*
* program sprawdza, czy nowe dane wprowadzane podczas edycji nie pokrywają się z danymi z istniejącej już bazy. Jeśli tak, to program wypisze komunikat: *Blad - w rejestrze znajduje sie juz taka osoba*, i nie zachowa wprowadzonych podczas edycji zmian,
* można podać dane maksymalnie 100 zawodników - taka jest pojemność tablicy.

**4. Przykładowe testy**

Wprowadzenie danych kilku zawodników i sprawdzenie możliwości edycji/usuwania poszczególnych elementów, jak również poprawności sortowania. Sprawdzenie, czy program działa poprawnie w przypadku podawania nieprawidłowych danych (właściwy komunikat, itd.).

**5. Schemat blokowy**

****