Politechnika Śląska w Gliwicach
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Temat projektu: Maraton

---

autor	Arkadiusz Pilarski
prowadzący	mgr inż. Marek Kokot
rok akademicki	2015/2016
kierunek	Informatyka
rodzaj studiów	stacjonarne
semestr	1
termin ćwiczeń	poniedziałek 10:00-11:30
grupa	1
sekcja	2
termin oddania pracy	czwartek, 29 stycznia 2016
data oddania pracy	poniedziałek, 25 stycznia 2016

---

1. Treść zadania

Maratończycy biorą udział w różnych zawodach. Po zakończeniu tworzona jest lista rankingowa w następującej postaci: w pierwszej linii jest nazwa maratoru, w drugiej data (w formacie: rrrr-mm-dd), w kolejnych są wyniki zawodników:
kolejność na mecie, nazwisko, nr zawodnika w zawodach, czas (w formacie: gg:mm:ss).
Przykładowy plik ma postać:

Maraton bostonski
2012-09-04
2, Jaworek, 1432, 04:34:12
1, Bukowy, 434, 03:54:45
3, Krol, 243, 04:37:32

Plików z wynikami zawodów może być dowolna liczba. Po wykonaniu programu uzyskujemy pliki wynikowe zawierające wyniki zawodników. Każdy plik zawiera wyniki tylko jednego maratończyka. Nazwa pliku jest tożsama z nazwiskiem maratończyka. W pliku tym podane jest nazwisko i wyniki, które uzyskał w zawodach. Wyniki te są przedstawione w następujący sposób: Data nazwa maratonu czas.
Wyniki są posortowane wg daty. Przykładowy plik dla Jaworka:

Jaworek

2011-05-03 Maraton Swiateczny 04:01:43
2011-09-14 Bieg Rzeznika 04:13:32
2012-05-03 Do przodu! 04:02:43

Program uruchamiany jest z linii poleceń z wykorzystaniem następującego przełącznika:
-i pliki wejściowe z protokołami zawodów.

2. Analiza zadania

Zagadnienie przedstawia problem tworzenia wielu plików z konkretnymi danymi na podstawie plików wejściowych, zawierających różne dane.

2.1 Struktury danych

W programie wykorzystano strukturę drzewa binarnego z podczepianymi listami jednokierunkowymi, których elementy wskazują na elementy osobnej listy jednokierunkowej.

Każdy węzeł drzewa przechowuje wskaźnik na węzeł na lewo i na prawo oraz wskaźnik na głowę podczepionej pod niego listy jednokierunkowej. Węzeł może mieć od 0 do 2 potomków, przy czym po lewej stronie węzła znajdują się potomki przechowujące wartości mniejsze niż węzeł rodzicielski, po prawej zaś większe (tymi wartościami są dane typu string, więc są one sortowane alfabetycznie).

Elementy listy jednokierunkowej, na której głowę wskazuje węzeł drzewa przechowują jedną wartość, wskaźnik na następny element tej listy, oraz wskaźnik na konkretny element drugiej listy jednokierunkowej.

Druga lista jednokierunkowa składa się z elementów, z których każdy zawiera dwie wartości, oraz wskaźnik na kolejny element listy.

Wszystkie elementy w strukturach użytych w programie są uporządkowane na podstawie odpowiedniej wartości.

2.2 Algorytmy

Program sortuje:

- nazwiska poprzez umieszczanie ich w drzewie binarnym;
- czasy konkretnych zawodników poprzez algorytm dodawania elementów do listy jednokierunkowej w porządku posortowanym wg daty zawodów;
- nazwy maratonów wraz z datami do drugiej listy jednokierunkowej w porządku posortowanym na podstawie daty zawodów;

3. Specyfikacja zewnętrzna

Program uruchamiany jest z linii poleceń z wykorzystaniem następującego przełącznika:

-i pliki wejściowe z protokołami zawodów

Pliki są plikami tekstowymi. Uruchomienie programu z parametrem –h powoduje wyświetlenie krótkiej pomocy. Uruchomienie programu z nieprawidłowymi parametrami powoduje wyświetlenie komunikatu: „Błędnie wprowadzone dane.” oraz krótkiej pomocy. Podanie nieprawidłowej nazwy pliku powoduje wyświetlenie odpowiedniego komunikatu: „Plik nie został otworzony poprawnie.” oraz ponownie krótkiej pomocy.

4. Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

4.1 Typy zdefiniowane w programie

W programie zdefiniowano następujące typy:

```
struct Zawodnicy
{
	string Nazwisko;
	Zawodnicy* lewy;
	Zawodnicy* prawy;
	Czas* Czas_zawodnika;
};
```

Ten typ służy do zbudowania drzewa binarnego.

```
struct Czas
{
	string time;
	Czas* next;
	Maratony* Zawody;

};
```

Ten typ służy do skonstruowania listy jednokierunkowej.

```
struct Maratony
{
	string nazwa_maratonu;
	string data;
	Maratony * next;
};
```

Ten typ służy do utworzenia osobnej listy jednokierunkowej.

4.2 Ogólna struktura programu ze szczegółowym opisem implementacji funkcji

W funkcji głównej wywoływana jest funkcja PobierzPliki :

```
void PobierzPliki(string* &pliki, int argc, char** argv);
```

Funkcja ta ma za zadanie pobrać nazwy plików przekazanych przez konsolę i umieścić je w dynamicznie utworzonej tablicy stringów.

Następnie wywoływana jest funkcja TworzenieStruktury w pętli. Funkcja ta wykona się tyle razy, ile plików zostało podanych przez konsolę.

```
for (int i = 0; i < argc - 2; i++)
            TworzenieStruktury(Nazwisko, czas, Data, tytul, plik, pliki[i], korzen,
glowa);

void TworzenieStruktury(string &Nazwisko, string &czas, string &Data, string &tytul,
ifstream &plik, string &nazwa_pliku, Zawodnicy* &korzen, Maratony* &glowa);
```

Funkcja ta ma za zadanie utworzenie struktury z danych pobranych z wszystkich przekazanych plików. W tej funkcji znajdują się trzy osobne funkcje: OtworzPlik, DodajDoListaMaratonow oraz DodajDoDrzewaZawodnikow.

```
void OtworzPlik(const string &nazwa_pliku, ifstream &plik);
```

Funkcja OtworzPlik otwiera plik z danymi. W razie nieudanej próby otwarcia wyświetla odpowiedni komunikat.

```
void DodajDoListaMaratonow(ifstream &plik, Maratony* &glowa, string &data, string
&tytul, string &Data, Maratony *&Maraton);
```

Funkcja DodajDoListaMaratonow pobiera z pliku nazwę maratonu oraz jego datę. Na tej podstawie tworzy listę jednokierunkową posortowaną wg daty maratonu.

```
void DodajDoDrzewaZawodnikow(ifstream &plik, Zawodnicy* &korzen, string &Nazwisko,
string &czas, Maratony* &Maraton, string &Data);
```

Funkcja DodajDoDrzewaZawodnikow pobiera z pliku nazwisko oraz czas konkretnego zawodnika i dodaje go do drzewa binarnego korzystając z funkcji Iteracyjnie.

```
void Iteracyjnie(Zawodnicy* &korzen, string &Nazwisko, Czas* Time, Maratony*
&Maraton, string &czas, string &Data);
```

Funkcja Iteracyjnie tworzy drzewo binarne dodając do niego węzły na podstawie parametrów przekazanych przez funkcję DodajDoDrzewaZawodnikow. Węzły drzewa są sortowane wg nazwisk zawodników. Elementem każdego węzła jest również wskaźnik na głowę listy jednokierunkowej, w której zapisywany jest czas zawodnika osiągnięty w danym maratonie. Każdy element tej listy przechowuje oprócz czasu, również wskaźnik na konkretny element listy tworzonej za pomocą funkcji DodajDoListaMaratonow, zawierający nazwę i datę maratonu, w którym zawodnik osiągnął ten konkretny czas.

Po utworzeniu struktury z pobranych danych wywoływana jest w funkcji głównej funkcja TworzPliki.

```
void TworzPliki(Maratony* &glowa, ofstream &zapis, Zawodnicy* &root);
```

Funkcja TworzPliki jest funkcją rekurencyjną, która przechodzi przez wszystkie węzły drzewa binarnego i na podstawie każdego węzła tworzy plik zawierający nazwisko oraz osiągniecia danego zawodnika uporządkowane wg daty maratonów, w których brał on udział.
W funkcji tej znajduje się również funkcja pomocnicza Dlugosc:

```
int Dlugosc(Maratony * &glowa);
```

Dzięki zwracanej przez nią wartości (długości najdłuższej nazwy maratonu) funkcja TworzPliki jest w stanie zapisać dane do pliku w sposób uporządkowany (data nad datą, nazwa maratonu nad nazwą maratonu, czas nad czasem).

Na samym końcu usuwana jest cała struktura, zwalniając pamięć.

```
void ZwolnijPamiec(Zawodnicy* &korzen, Maratony* &glowa);
```

Funkcja ZwolnijPamiec powoduje wywołanie dwóch osobnych funkcji, z których pierwsza (UsunDrzewoZCzasami) usuwa strukturę drzewa binarnego z podczepianymi listami jednokierunkowymi, a druga (UsunListeMaratonow) usuwa listę jednokierunkową zawierającą nazwy i daty maratonów.

```
void UsunDrzewoZCzasami(Zawodnicy* &root);

void UsunListeMaratonow(Maratony * &glowa);
```

5. Testowanie

Program został przetestowany na różnego rodzaju plikach. W przypadku niepoprawnego wywołania programu (niewłaściwe parametry) wyświetlane są odpowiednie komunikaty wraz z pomocą jak poprawnie uruchomić program. W wypadku podania pliku pustego, program nie dodaje do struktury żadnych danych i kontynuuje pracę.

5. Wnioski

Praca nad tym programem pozwoliła mi lepiej zrozumieć pojęcia takie jak lista czy drzewo. Nauczyła mnie także dbania o szczegóły i pamiętania o detalach podczas tworzenia funkcji. Popracowałem również nad wyszukiwaniem błędów w programie. Dowiedziałem się też, że należy unikać zmiennych globalnych – lepiej tworzyć zmienne lokalne i przekazywać je do kolejnych funkcji.