Sprawozdanie z laboratorium

Tadeusz Małuszyński

14 kwietnia 2015

1: Zadanie

Zaimplementować tablicę mieszającą wraz z funkcją haszującą. Przedyskutować złożoność funckji benchmarkującej i uzasadnić jej wybór.

2: Implementacja

Korzystając z zaimplementowanej na laboratorium 2 struktury danych typu dllist (lista dwustronnie łączona) zaimplementowano tablicę haszującą na podstawie funkcji h() przetwarzającej obiekty typu string w obiekty typu int. Zaimplementowana funkcja oddaje liczbę od 0 do 1023, przez co tablica ma 1024 pola.
Każde pole tablicy jest osobnym obiektem klasy dllist, tablica zaś jest deklarowana jako statyczna tablica 1024 wskaźników na dllist. Dzięki takiemu rozwiązaniu kolizje są rozwiązywane poprzez dokładanie nowych elementów do listy w razie, gdyby hasz się powtórzył.
Zastosowano bardzo prostą funkcję haszującą: Otrzymuje ona stringa, tworzy liczbę typu int o wartości 0, bierze wartość ASCII jednego znaku a następnie mnoży tą wartość przez podaną liczbę pierwszą (w tym konkretnym przypadku 23), dodaje do posiadanej liczby, po czym wyciąga z wyniku modulo 1024. Następnie operacja ta powtarzana jest dla każdego kolejnego znaku, otrzymana ostatecznie liczba jest używana jako indeks elementu w którym umieszczony zostanie element.
Program testujący tablicę liczy ilość razy którą wystąpiło w podanym pliku txt każde występujące w nim słowo, a następnie drukuje całą tablicę z pominięciem pustych elementów. Liczba wypisana na konsoli obok słowa podaje liczbę wystąpień w tekście. W związku z właściwościami operatora string«fstream do stringów są zainkorporowane wszystkie znaki występujące w kodzie ASCII, znaki białe enter i spacja powodują zakończenie aktualnie wczytywanego stringa. Dałoby się to prawdopodobnie obejść przy pomocy metody getchar() ale spowodowałoby to wzrost zapotrzebowania na surowce, gdyż każdy string wymagałby osobnej pętli.

3. Zastosowane rozwiązania

Zastosowana funkcja haszująca jest niezwykle szybka - jej złożoność jest rzędu O(n), gdzie n jest długością otrzymanego stringa. Jest ona również niestety dość podatna na kolizje, przykładowo jest w pewnym stopniu niezależna od kolejności liter. Zastosowano tą funkcję ze względu na prostotę implementacji i szybkość działania - o ile można by zastosować funkcję mniej podatną na kolizje i dającą lepsze wyniki, ale prowadziłoby to do nadmiernych komplikacji w implementacji przy stosunkowo niewielkich korzyściach.
Wykorzystano napisaną wcześniej klasę Logger która pozwala na zapisanie stringów do osobnego pliku .txt.

4. Uwagi

Nie znaleziono przyczyny pojawiania się przy niektórych kompilacjach dziwnego znaku na początku dokumentu i błędu SEGFAULT. Wykorzystanie debuggera w tym przypadku w niczym nie pomogło, nadal niemożliwym wydaje się być odnalezienie źródła problemu.

5. Wykorzystane narzędzia

Do napisania programu wykorzystano IDE Eclipse z kompilatorem MinGW32 i cmake'a. Do napisania sprawozdania wykorzystano środowisko LaTeX.