Wstęp do programowania w C

Robert Ferens

Lista 6 - 17 listopada 2021

Zadanie 1

Zdefiniuj typ hash_t za pomocą typedef jako synonim na long long unsigned int (1pkt) i stwórz funcję haszującą(1pkt):

```
hash_t hash(char word[]);
```

Funkcja powinna zwrócić pewną liczbę z przedziału [0,MAX_HASH] na podstawie zadanego słowa. Dobra funkcja haszująca ma tę własność, że szansa że dwa różne słowa dadzą w wyniku tę samą liczbę jest niska(co nie znaczy że nie istnieje) - tzn. jest odporna na kolizję. (w tym zadaniu nie zależy nam na własności jednokierunkowości - zamazaniu informacji o argumencie funkcji haszującej na podstawie hasza)

Zaimplementuj następujący algorytm funkcji haszującej(4pkt):

1. Ustaw zmienną h na 0
2. Weź pierwszą nie rozważaną dotąd literę słowa
3. Pomnóż h przez 257
4. Dodaj do niego wartość liczbową pobranej litery
5. Ustaw jako h, h modulo MAX_HASH
6. Wróć do kroku 2, lub przejdź dalej jeśli słowo się skończyło
7. Zwróć h

(2pkt) Policz ilość wystąpień hashów słów w tekście (gdyby funkcja haszująca nie miala kolizji byłoby to jednoznaczne z policzeniem ilości wystąpień każdego słowa). Słowo traktujemy jako ciąg znaków oddzielony znakami białymi(jest to definicja zgodna z wczytywanieniem słów za pomocą $scanf(''\%s'', s)$). Załóż że słowa w danych będą krótsze niż 60 znaków.

(2pkt) Wypisz ilość wystąpień dla każdego hasza gdy MAX_HASH=100, a na wejście podamy tekst Pana Tadeusza(dołączony na skos). Czy jest to rozkład bliski jednorodnemu?

PS. Dla ciekawych: jeśli kogoś interesuje dla zadanej liczby hashy ile wartości można pomieścić tak, żeby szansa kolizji wynosiła mniej niż 50% to warto poczytać o paradoksie urodzin i skorzystać z wyliczonego wzoru by to oszacować: `https://www.wikiwand.com/pl/Paradoks_dnia_urodzin`

Zadanie 2

Skorzystaj z napisanej w poprzednim zadaniu funkcji by policzyć ilość różnych słów występujących w Panu Tadeuszu, oraz najczęściej występującego słowa. Napisz tablicę

```
char words[MAX_HASH][60]
```

zapisującą pod wybranym numerem zwróconym przez funkcję pierwsze napotkane słowo o danym haszu, aby można było odwrócić haszowanie.

Za pomocą strncmp policz ilość napotkanych kolizji(odrzuć próby nadpisania już zajętego haszu innym słowem) i zobacz jak zależy ona od wielkości MAX_HASH. Następnie policz w innej tablicy ilość wystąpień tych słów i wypisz najczęściej występujące.

UWAGA! Dążenie do wartości MAX_HASH dla których już nie występują kolizje może spowodować przekroczenie ilości dostępnej pamięci RAM na tablicę words, zobacz coś się wtedy stanie. Potem wróć do wartości które są odpowiednie dla komputera i zignoruj słowa które kolidują z pierwszymi zapisanymi w tablicy.

UWAGA2! Kopiowanie stringu do tablicy nie może odbyć się za pomocą words[i] = inp_word - spowoduje to wyciek pamięci - szczegóły dla zainteresowanych na ćwiczeniach. Stringi kopiuje się za pomocą funkcji strcpy/strncpy lub ręcznie znak po znaku.

PS. Dla ciekawych takie liczenie haszów stosuje sie w praktyce w wielu miejscach (hash map/hash table), jednak nie żąda się od funkcji braku kolizji, lecz w każdym miejscu tablicy podpina się strukturę danych trzymającą wszystkie reprezentacje kolidujące. Jednak do osiągniecia tego efektu należałoby skorzystać ze struktury listy(wskaźniki) lub realokacji pamięci dla zmiany wielkości tablicy (wskaźniki i zarządzanie pamięcią realloc) co wykracza ponad poznany do tej pory materiał.

Zadanie 3

Dostępne ze sprawdzaczką na skosie.