1. Hash table with separate chaining

1 Uwagi ogólne

Celem ćwiczenia jest zaimplementowanie struktury opisującej tablicę z haszowaniem elementów dowolnego typu. Do rozwiązywania problemu kolizji zastosujemy metodę łańcuchową. Należy zaimplementować opisane niżej funkcje typowe dla tej struktury danych.

Ponieważ alokowanie pojedynczych elementów typów prymitywnych jest nie jest rozwiązaniem optymalnym, jako element danych zastosowano unię, składającą się z pól odpowiadających wykorzystywanym typom prymitywnym oraz wskaźnika używanego w przypadku elementu będącego strukturą. W ten sposób typy prymitywne są zawarte bezpośrednio w strukturze elementu a alokacja zachodzi tylko dla typów strukturalnych.

W kolejnych punktach na podstawie stworzonego szablonu będziemy tworzyć tablice liczb całkowitych, znaków, oraz struktur `DataWord` (jak w templatce programu), składających się ze słowa i licznika.

2 Struktura tablicy i funkcje ogólne

Struktura `hash_table` składa się z:

1. wskaźnika do początku (dynamicznie alokowanej) tablicy przechowującej głowy list (sentinels) elementów o jednakowej wartości funkcji haszującej
2. rozmiaru tablicy mieszającej
3. aktualnej liczby elementów zapisanych w tablicy
4. wskaźników do funkcji odpowiadających za operacje związane z konkretnym typem elementów (wypisywanie elementu, zwalnianie pamięci elementu, wczytanie / utworzenie danych określonego typu, komparator, funkcja haszująca, funkcja modyfikująca dane w przypadku próby wpisania do tablicy istniejącego klucza)

Szablon programu należy uzupełnić o definicję następujących funkcji:

1. `init_ht()` – alokuje tablicę na zadaną początkową długość i inicjalizuje pozostałe pola.
2. `dump_list()` – wypisuje listę elementów związaną z daną wartością funkcji haszującej.
3. `free_element()` – zwalnia pamięć danych i elementu tablicy.
4. `free_table()` – zwalnia pamięć wszystkich elementów tablicy (a także samą tablicę).
5. `insert_element()` – dodaj element do tablicy.

6. `rehash()` – zwiększ dwukrotnie rozmiar tablicy i rozmieść elementy zgodnie z nowymi wartościami funkcji haszującej. **Uwaga**: funkcja pobiera kolejne elementy ze "starej" tablicy i przepina je do "nowej" dodając je na początek właściwej listy
7. `remove_element()` – usuń element o zadanym kluczu
8. `get_element()` – zwróć adres elementu o zadanym kluczu

Uwaga: Jeżeli po dodaniu elementu do tablicy (funkcja `insert_element()`) współczynnik wypełnienia tablicy (stosunek liczby elementów tablicy do jej długości) przekroczy zadaną wartość `MAX_RATE` długość tablicy jest zwiększana dwukrotnie i wołana jest funkcja `rehash()`.

Ogólna postać danych (do podpunktu 1 i 2):

numer zadania
`n` – liczba komend
`index` – indeks tablicy, którego elementy należy wydrukować
`n` linii komend

Każda komenda składa się z litery (kodu komendy) i pozostałych danych (w zależności od typu polecenia).

W przypadku próby dodania istniejącego klucza element nie jest dodawany.

Lista komend:

1. `i value - insert_element(value)`
2. `r value - remove_element(value)`

Wyjściem z każdej sekcji jest długość tablicy i elementy należące do listy o numerze `index`.

2.1 Tablica liczb całkowitych

Dodatkowo szablon programu należy uzupełnić o funkcje:

1. `hash_int()` – oblicz wartość funkcji haszującej dla parametru typu`int`
2. `dump_int()` – wypisz element typu `int`
3. `cmp_int()` – komparator wartości całkowitych
4. `create_int()` – wczytaj wartość całkowitą. Funkcja zwraca unię `data_union` zawierającą wczytaną wartość. Dodatkowo, jeżeli parametr nie jest wskaźnikiem `NULL`, to powinien zawierać adres unii `data_union`, do której zostanie wpisana wczytana liczba.

Przykład:

Wejście:

```
1
10 3
i 12345
i 1334
```

```
i 1534
i 3234
i 5234
i 7234
i 8234
i 9234
r 5234
i 234
```

Wyjście:

```
4
9234 8234 7234
```

2.2 Tablica znaków

Dodatkowo szablon programu należy uzupełnić o funkcje:

1. `hash_char()` – oblicz wartość funkcji haszującej dla parametru typu`char`
2. `dump_char()` – wypisz element typu `char`
3. `cmp_char()` – komparator wartości znakowych
4. `create_char()` – wczytaj znak. Funkcja zwraca unię `data_union` zawierającą wczytaną wartość. Dodatkowo, jeżeli parametr nie jest wskaźnikiem `NULL`, to powinien zawierać adres unii `data_union`, do której zostanie wpisany wczytany znak.

Przykład:

Wejście:

```
2
12 2
i C
i i
i t
i M
r C
i u
i P
i l
i f
i W
r o
i s
```

Wyjście:

```
4
l M t
```

2.3 Tablica struktur `DataWord`

Dodatkowo szablon programu należy uzupełnić o funkcje:

1. `hash_word()` – oblicz wartość funkcji haszującej dla parametru typu `DataWord`
2. `dump_word()` – wypisz element typu `DataWord` (słowo i licznik)
3. `free_word()` – zwolnik pamięć słowa i elementu typu `DataWord`
4. `cmp_word()` – komparator wartości typu `const char*` (słów)
5. `modify_word()` – zwiększ wartość licznika zadanego elementu
6. `create_data_word()` – utwórz i zwróć unię `data_union` ze słowem zadanym parametrem funkcji i licznikiem równym 1
7. `stream_to_ht()` – czytaj linie tekstu, wyodrębnij słowa (zdefiniowane jak w zadaniu z listą jednokierunkową) i dodaj do tablicy po zamianie na małe litery; przy powtarzających się kluczach (słowach) zwiększ licznik słowa (funkcja `modify_data()`

Wejście

3
`word` – słowo, które należy wypisać
linie tekstu

Wyjście

długość tablicy
słowo (podane na wejściu) i jego liczność

Przykład:

Wejście:

```
3
xyz
abcd xyz ab;XYz qwerty
```

Wyjście:

```
8
xyz 2
```