ЛАБОРАТОРНА РОБОТА 3
Таблиця

ТЕОРЕТИЧНИЙ МАТЕРІАЛ

Структура даних “проста таблиця” повинна зберігати колекцію пар ключ-значення та реалізувати доступ до значень за ключем.

Ключі потрібно шукати швидко, значення розташовувати в пам'яті просто.

Основні операції над таблицею:

- включення в таблицю нової пари (так, щоб можна було швидко знайти значення за ключем);
- заміна в таблиці за ключем старого значення на нове;
- вилучення з таблиці пари за заданим ключем;
- пошук значення в таблиці за ключем;
- роздрук всієї таблиці (всіх пар, що вона містить).

Додаткові операції:

- отримання колекції (переліку) ключів;
- отримання колекції значень;
- перебір ключів, значень, пар для виконання довільних дій.

Рекомендації стосовно програмної реалізації:

1. Використати динамічні масиви для зберігання ключів і значень. За потреби можна налаштувати їхній розмір.
2. Ключі зберігати у впорядкованому масиві (забезпечить швидкість пошуку). Тобто, додавання/вилучення ключа перебудовуватиме масив.
3. Значення зберігати у звичайному масиві (забезпечить простоту розміщення). Вилучення ототожнюється встановленням відповідної позначки.
4. Час від часу потрібне буде ущільнення масиву значень - остаточне вилучення позначених записів.

Зразок шаблону заголовного файлу для реалізації структури “проста таблиця”:

```
#include <string>
#include <iostream>
using namespace std;

class Table {
public:
  typedef int KeyType; // тип ключів
  typedef string ValueType; // тип значень
```

```
  struct KeyNode {
    KeyType key; // ключ
    int index; // індекс значення в масиві значень
  };
  struct ValueNode {
    ValueType value; // значення
    int index; // індекс ключа в масиві ключів
  };

private:

  int size; // розмір таблиці
  int count; // кількість записів
  int place; // номер першої вільної комірки в масиві значень
  KeyNode* keys;
  ValueNode* values;

  void resize(); // збільшення розміру та перерозподіл пам'яті

  // стискання масиву значень (видалення позначених елементів)
  void press();

  // пошук ключа або місця для нього
  bool find(KeyType key, int& index);

  // запис значення в знайденому місці
  void store(KeyType key, const ValueType& value, int i);

public:

  Table(int startSize); // конструктор
  ~Table(); // деструктор

  int getCount() {return count;}

  // включення нової пари або заміна значення за ключем
  Table& put(KeyType key, const ValueType& value);

  // вилучення пари за заданим ключем
  Table& erase(KeyType key);

  // існування значення за заданим ключем
  bool contain(KeyType key);

  ValueType getValue(KeyType key);
  KeyType getKey(const ValueType& value);
```

```
  ValueType& operator[](KeyType key);

  void print(); // виведення записів у консоль

  KeyType* getKeys();
  ValueType* getValues();

  // перебір ключів, значень, пар для виконання дій
  void keysDo(void (*KeyDelegate) (KeyType));
  void valuesDo(void (*ValueDelegate) (ValueType&));
  void Do(void (*PairDelegate) (KeyType, ValueType&));
};
```

Зразок демонстрації структури:

```
void printPair(Table::KeyType key, Table::ValueType& value) {
      cout << value << "'s code is " << key << endl;}

int main() {
  cout << "-------------" << endl;
  Table T(3); T.print();
  cout << "Table is created. Number of entries: " << T.getCount() << endl;
  T.put(380, "Ukraine");
  T.put(39, "Italy").put(34, "Spain");
  T.print();
  cout << "---Number of entries: " << T.getCount() << endl;
  T.erase(39).print();
  cout << "---Number of entries: " << T.getCount() << endl;
  T.put(39, "Italy").put(52, "Mexico").print();
  cout << "---Table is resized. Number of entries: " << T.getCount() << endl;
  cout << "-------------" << endl;
  cout << "Does the table contain the key [40]? " <<   T.contain(40) << endl;
  cout << "Does the table contain the key [39]? " <<   T.contain(39) << endl;
  cout << "The value at key [380] is " << T.getValue(380) << endl;
  cout << "The key for value [Ukraine] is " << T.getKey("Ukraine") << endl;
  cout << "-------------" << endl;
  cout << "Adding the pair 1-->USA by operator []:" << endl;
  T[1] = "USA"; T.print();
  cout << "Getting the value by operator []: " << T[380]<< endl;
  cout << "-------------" << endl;
  Table::KeyType* Keys = T.getKeys();
  cout << "KEYS: ";
  for (int i=0; i<T.getCount(); ++i) {cout << Keys[i] << ' ';}
  cout << endl;
  Table::ValueType* Values = T.getValues();
  cout << "VALUES: ";
  for (int i=0; i<T.getCount(); ++i) {cout << Values[i] << ' ';}
  cout << endl << "---Sending pairs to function---" << endl;
  T.Do(printPair);
  return 0;
}
```

```
mokasin@mokasin-Inspiron-15-3573:~/Документи/ДИСЦИПЛІНИ/АЛГОРИТМИ І СТРУКТУРИ/ТАБЛИЦЯ/CPP$ ./m
-------------
Table is empty
Table is created. Number of entries: 0
34 --> Spain
39 --> Italy
380 --> Ukraine
---Number of entries: 3
34 --> Spain
380 --> Ukraine
---Number of entries: 2
34 --> Spain
39 --> Italy
52 --> Mexico
380 --> Ukraine
---Table is resized. Number of entries: 4
-------------
Does the table contain the key [40]? 0
Does the table contain the key [39]? 1
The value at key [380] is Ukraine
The key for value [Ukraine] is 380
-------------
Adding the pair 1-->USA by operator []:
1 --> USA
34 --> Spain
39 --> Italy
52 --> Mexico
380 --> Ukraine
Getting the value by operator []: Ukraine
-------------
KEYS: 1 34 39 52 380
VALUES: Ukraine Spain Italy Mexico USA
---Sending pairs to function---
USA's code is 1
Spain's code is 34
Italy's code is 39
Mexico's code is 52
Ukraine's code is 380
mokasin@mokasin-Inspiron-15-3573:~/Документи/ДИСЦИПЛІНИ/АЛГОРИТМИ І СТРУКТУРИ/ТАБЛИЦЯ/CPP$
```

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Реалізувати засобами C++ структуру “проста таблиця”, взявши до уваги наведені вище рекомендації та перелік операцій. Продемонструвати використання структури на конкретному прикладі, згідно з Вашим варіантом. У варіантах вказані описи пар ключ-значення.

ВАРІАНТИ ЗАВДАНЬ ДЛЯ ІНДИВІДУАЛЬНОГО ВИКОНАННЯ:

1. Назва товару - Ціна.
2. Логін - Пароль.
3. Особа - Ріст.
4. Номер студентського квитка - Прізвище та ініціали студента.
5. Назва дисципліни - Оцінка.
6. Місто - Відстань від Києва.

7. Країна - Площа території.
8. Телепередача - Час початку.
9. Пісня - Перший виконавець.
10. Номер телефону - Тариф.
11. Поштовий індекс - Населений пункт.
12. Столиця - Країна.
13. Гора - Висота.
14. Книга - Автор.
15. Спортсмен - Особистий рекорд зі стрибків у довжину.
16. Код товару - Назва товару.
17. Ім’я файлу - Розмір.
18. Промокод - Сума знижки.
19. Фільм - Тривалість.