**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра АПУ**

**ОТЧЕТ**

**по лабораторной работе №5**

**по дисциплине «Программирование»**

| Студент гр. 3391 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Николаев В.Ю. |
| --- | --- | --- |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Писарев А.С. |

Санкт-Петербург

2024 г.

# Исходная формулировка

Разработать программу обеспечивающую:

1. формирование хэш-таблицы (размерность – не более числа литер в латинском алфавите), позволяющей хранить информацию о введенных с клавиатуры словах; механизм обработки коллизий в построенной таблице должен обеспечиваться путем формирования вспомогательных списков «конфликтующих» слов, связываемых с соответствующей ячейкой таблицы;
2. реализацию поиска заданного слова в хэш-таблице.

# Ход работы программы

Программа начинается с определения структуры WordInfo, которая содержит слово и вектор конфликтующих слов. Затем инициализируется вектор hashTable размером MAX\_SIZE (26), который представляет собой хеш-таблицу.

| 8  9  10  11  12  49 | struct WordInfo  {  std::string word;  std::vector<std::string> conflicts;  };  std::vector<WordInfo> hashTable(MAX\_SIZE); |
| --- | --- |

Функция hashFunction принимает слово и возвращает хеш, который является индексом в хеш-таблице. Это просто ASCII-значение первого символа слова минус 'a'.

| 14  15  16  17 | int hashFunction(const std::string& word)  {  return word[0] - 'a';  } |
| --- | --- |

Функция insertWord принимает слово, вычисляет его хеш и добавляет его в соответствующую ячейку в хеш-таблице. Если ячейка уже занята, слово добавляется в вектор конфликтов.

| 19  20  21  22  23  24  25  26  27 | void insertWord(std::vector<WordInfo>& hashTable, const std::string& word)  {  int index = hashFunction(word);    if (hashTable[index].word.empty())  hashTable[index].word = word;  else  hashTable[index].conflicts.push\_back(word);  } |
| --- | --- |

Функция searchWord принимает слово, вычисляет его хеш и ищет его в соответствующей ячейке в хеш-таблице. Если слово найдено, функция возвращает true, в противном случае - false.

| 29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | bool searchWord(const std::vector<WordInfo>& hashTable, const std::string& word)  {  int index = hashFunction(word);    if (hashTable[index].word.empty())  return false;  else if (hashTable[index].word == word)  return true;  else  {  for (const std::string& conflict : hashTable[index].conflicts)  if (conflict == word)  return true;  return false;  }  } |
| --- | --- |

В функции main реализован простой текстовый интерфейс, который позволяет пользователю вставлять и искать слова.

| 47  48  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  87  88  89  90  91  92  93  94 | int main()  {  // ...  do  {  std::cout << "Menu:\n";  std::cout << "1. Insert word\n";  std::cout << "2. Search word\n";  std::cout << "3. Exit\n";  std::cout << "Enter your choice: ";  std::cin >> choice;    switch (choice)  {  case 1:  std::cout << "Enter word to insert: ";  std::cin >> word;  insertWord(hashTable, word);  break;  case 2:  std::cout << "Enter word to search: ";  std::cin >> word;  if (searchWord(hashTable, word))  std::cout << "Word found\n";  else  std::cout << "Word not found\n";  break;  // ...  }    std::cout << std::endl;  }  while (choice != 3);    return 0;  } |
| --- | --- |

# Текст программы

| Main.cpp | |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94 | #include <iostream>  #include <string>  #include <vector>  const int MAX\_SIZE = 26;  struct WordInfo  {  std::string word;  std::vector<std::string> conflicts;  };  int hashFunction(const std::string& word)  {  return word[0] - 'a';  }  void insertWord(std::vector<WordInfo>& hashTable, const std::string& word)  {  int index = hashFunction(word);    if (hashTable[index].word.empty())  hashTable[index].word = word;  else  hashTable[index].conflicts.push\_back(word);  }  bool searchWord(const std::vector<WordInfo>& hashTable, const std::string& word)  {  int index = hashFunction(word);    if (hashTable[index].word.empty())  return false;  else if (hashTable[index].word == word)  return true;  else  {  for (const std::string& conflict : hashTable[index].conflicts)  if (conflict == word)  return true;  return false;  }  }  int main()  {  std::vector<WordInfo> hashTable(MAX\_SIZE);    int choice;  std::string word;    do  {  std::cout << "Menu:\n";  std::cout << "1. Insert word\n";  std::cout << "2. Search word\n";  std::cout << "3. Exit\n";  std::cout << "Enter your choice: ";  std::cin >> choice;    switch (choice)  {  case 1:  std::cout << "Enter word to insert: ";  std::cin >> word;  insertWord(hashTable, word);  break;  case 2:  std::cout << "Enter word to search: ";  std::cin >> word;  if (searchWord(hashTable, word))  std::cout << "Word found\n";  else  std::cout << "Word not found\n";  break;  case 3:  std::cout << "Exiting...\n";  break;  default:  std::cout << "Invalid choice\n";  break;  }    std::cout << std::endl;  }  while (choice != 3);    return 0;  } |

# Результаты работы программы