МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е. В. Павлов |
| (должность, уч. степень, звание) |  | (подпись, дата) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

«ОПРЕДЕЛЕНИЕ СЛОЖНОСТИ ПРОГРАММНОГО КОДА

НА ОСНОВЕ ЛЕКСИЧЕСКОГО АНАЛИЗА»

ПО КУРСУ: «МЕТРОЛОГИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

РАБОТУ ВЫПОЛНИЛ:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4831 |  |  |  | 22.03.2020 |  | К.А. Корнющенков |
|  |  |  | (подпись) |  | (дата отчёта) |  | (инициалы, фамилия) |

Санкт-Петербург 2020

1. Цель работы

Целью данной работы является оценка сложности программного кода на основе лексического анализа с использованием метрик Холстеда и Джилба.

1. Задание на лабораторную работу

Составить словарь операторов и операндов программного кода приложения (или его фрагмента) и произвести расчёт метрик Холстеда и Джилба.

Выполнить анализ полученных результатов.

Разрешается ограничить исходный код программы и использовать для расчётов метрик Холстеда и Джилба фрагмент кода (или модуль) размером 160-220 строк (из которых не более 15% пустых).

|  |
| --- |
| Вариант задания:  89. Приложение для обмена фотографиями |

1. Словарь операторов и операндов

Анализируемый программный код представлен в *Приложении А*.

Таблица 1 — Словарь операторов

|  |  |  |  |
| --- | --- | --- | --- |
| N | Операторы | Номера строк | Количество повторений |
| 1 | {} | 1(48),4(47),16(46),50(58),60(95),65(91),66(90),73(88),79(81),82(84),92(94),97(104),98(100),101(103),106(113),107(109),110(112),115(156),121(155),122(137),127(132),128(131),138(151),144(147),147(149),158(172),160(162),174(185),176(178),187(210),189(207),190(195),191(194),196(206),198(200),202(204),207(209),212(216) | 38 |
| 2 | << | 5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13,14,14,52,52,54,54,57,57,63,63,68,68,69,69,70,70,76,76,80,80,80,80,83,83,93,93,99,99,99,99,102,102,108,108,111,111,116,116,117,117,118,118,125,125,135,135,140,140,142,142,146,146,148,148,153,153,159,159,161,161,161,161,161,161,165,165,166,266,166,166,168,168,171,171,175,175,177,177,177,177,177,177,181,181,182,182,182,182,184,184,197,197,197,197,199,199,201,203,203,203,203,205,208,208,213,213,214,214,215,216 | 128 |
| 3 | >> | 15,53,55,64,72,120,126,141,143,164,169,180 | 12 |
| 4 | ; | 2,3,5,6,7,8,9,10,11,12,13,14,15,18,19,21,22,24,25,27,28,30,31,33,34,36,37,39,40,42,43,45,51,52,53,54,55,56,57,61,62,63,64,65,65,67,68,69,70,71,72,75,76,77,79,79,80,83,85,87,89,93,98,98,99,102,107,107,108,111,116,117,118,119,120,123,124,125,126,127,127,129,130,133,134,135,136,139,140,141,142,143,145,146,148,150,153,154,159,160,160,161,163,164,165,166,167,168,169,170,171,175,176,176,177,179,180,181,182,183,184,188,190,190,192,193,196,196,197,198,198,199,201,202,202,203,205,208,213,214,215 | 141 |
| 5 | if | 66,82,92,101,110,128,144,189,191 | 9 |
| 6 | else | 147,207 | 2 |
| 7 | = | 2,4,62,65,67,79,98,107,124,127,129,133,134,145,160,170,176,190,192,196,198,202 | 24 |
| 8 | == | 66,82,101,110,128,144,191 | 7 |
| 9 | while | 4 | 1 |
| 10 | for | 65,79,98,107,127,160,176,190,196,198,202 | 11 |
| 11 | ! | 4,92,199 | 3 |
| 12 | + | 65,65,79,79,98,98,107,107,127,127,160,160,161,176,176,177,183,190,190,196,196,198,198,202,202 | 25 |
| 13 | - | 2,166,166,170,182,182,183 | 7 |
| 14 | int | 2,65,71,79,98,107,119,124,127,160,163,176,179,190,196,198,202 | 17 |
| 15 | string | 61,123,139,167 | 4 |
| 16 | switch | 16,73,121 | 3 |
| 17 | case | 17,20,23,26,29,32,35,38,41,74,78,122,138 | 13 |
| 18 | break | 19,22,25,28,31,34,37,40,43,45,77,85,87,89,130,136,150,154,193 | 19 |
| 19 | () | 1,18,21,24,27,30,33,36,39,42,50,60,65,79,82,97,98,101,106,107,110,115,127,158,160,174,186,183,187,189,190,196,198,202,212 | 35 |
| 20 | default | 44,86,152 | 3 |
| 21 | bool | 62 | 1 |
| 22 | system | 3 | 1 |
| 23 | Photo | 51 | 1 |
| 24 | ofstream | 188 | 1 |
| 25 | void | 1,50,60,97,106,115,158,174,187,212 | 10 |
| *Итого:* | | | 516 |

Таблица 2 — Словарь операндов

|  |  |  |  |
| --- | --- | --- | --- |
|  | Операнды | Номера строк | Количество повторений |
| 1 | cout | 5,6,7,8,9,10,11,12,13,14,52,54,57,63,68,69,70,76,80,83,93,99,102,108,111,116,117,118,125,135,140,142,146,148,153,159,161,165,166,168,171,175,177,181,182,184,208,213,214,215 | 50 |
| 2 | cin | 15,53,55,64,72,120,126,141,143,164,169,180 | 12 |
| 3 | choose | 2,4,15,16,71,72,73,119,120,12 | 10 |
| 4 | photo | 51,53,55,56 | 4 |
| 5 | login | 61,64,66,123,126,133 | 6 |
| 6 | find | 62,67,92 | 3 |
| 7 | fasle | 62 | 1 |
| 8 | true | 67 | 1 |
| 9 | index | 124,129,134,163,164,166,166,170,179,180,182,182,183 | 13 |
| 10 | password | 139,141,144,145 | 4 |
| 11 | password2 | 139,143,144 | 3 |
| 12 | description | 167,169,170 | 3 |
| 13 | file | 188,189,197,199,201,203,205 | 7 |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 |  |  |  |
| 17 |  |  |  |
| 18 |  |  |  |
| 19 |  |  |  |
| 20 |  |  |  |
| 21 |  |  |  |
| 22 |  |  |  |
| 23 |  |  |  |
| *Итого:* | | |  |

1. Оценка сложности программного кода

Таблица 3 — Значения метрик Холстеда

|  |  |  |
| --- | --- | --- |
| Наименование характеристики | Обозначение и формула  для вычисления | Значение |
| Число простых (уникальных) операторов |  |  |
| Число простых (уникальных) операндов |  |  |
| Общее число всех операторов |  |  |
| Общее число всех операндов |  |  |
| Словарь программы |  |  |
| Длина реализации программы |  |  |
| Расчётная длина программы |  |  |
| Объём программы (в битах) |  |  |
| Сложность программы |  |  |
| Усилие |  |  |
| Время реализации (в секундах) |  |  |

Таблица 4 — Значения метрики Джилба

|  |  |  |
| --- | --- | --- |
| Наименование характеристики | Обозначение и формула  для вычисления | Значение |
| Абсолютная сложность программы |  |  |
| Общее количество операторов в программе |  |  |
| Относительная сложность программы |  |  |

Выводы по работе

В результате выполнения данной работы произведены расчёты показателей сложности для программного кода (см. Приложении А) с использованием метрик Холстеда и Джилба. Согласно заданию расчёт указанных метрических характеристик необходимо выполнить ручным методом, для этих целей был составлен словарь операторов и операндов анализируемого программного кода.

При анализе полученных значений метрик можно выделить следующее:

* Значение длины реализации больше значения расчетной длины на 27%, что может свидетельствовать о неудачных приёмах кодирования или избыточности при реализации программного модуля, так как согласно метрике Холстеда разница между двумя указанными характеристиками должна быть в пределах 10%. Анализ причин выявленного несоответствия заданному нормативному значению выходит за рамки данной работы;
* Время реализации *T* (около 5 часов) близко к реальному времени, которое было затрачено на кодирование анализируемого фрагмента кода;
* Необходимо отметить, что показатель сложности программы следует рассматриваться вкупе с аналогичными расчетами по другим проектам или в рамках текущего проекта для идентичных задач. В контексте настоящей работы данный показатель сложно интерпретировать с точки зрения оценки трудоёмкости кодирования или сложности проверки кода;
* Анализируемый программный код обладает невысокой относительной сложностью (*cl* = 0,054), так как на 493 оператора приходится 4 циклические конструкции и 23 оператора условий, что позволяет говорить о низкой логической сложности данного программного кода.

Таким образом, можно заключить, что выполненная работа соответствует поставленной задаче и отвечает всем сформулированным в методических указаниях требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Черников, Б. В. Управление качеством программного обеспечения: учебник /   
   Б. В. Черников. — М.: ИД «ФОРУМ»: ИНФРА-М, 2012. — 240 с.: ил.
2. Широков, А. И. Стандартизация, сертификация и оценка качества программного обеспечения: учебное пособие / А. И. Широков, Е. П. Потоцкий. —   
   М.: ИД «МИСиС», 2013. — 208 с.
3. Справочное руководство по C++ (машинный перевод) [Электронный ресурс]: Microsoft Docs. — Microsoft, 2020. — URL: <https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=vs-2017> (дата обращения: 17.03.2020).
4. Halstead Metrics [Электронный ресурс]: Static Metrics for C, C++, Ada and Java. — IBM Knowledge Center, 2020. — URL: <https://www.ibm.com/support/knowledgecenter/SSSHUF_8.0.0/com.ibm.rational.testrt.studio.doc/topics/csmhalstead.htm> (дата обращения: 17.03.2020).

ПРИЛОЖЕНИЕ А

Анализируемый программный код

|  |  |
| --- | --- |
| 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  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250 | void Instagramm::menu(){  int choose = -1;  system("cls");  while (choose!=8) {  cout << "Меню" << endl;  cout << "1 - просмотр своих фотографий" << endl;  cout << "2 - добавление фотографий " << endl;  cout << "3 - удаление своих фотографий " << endl;  cout << "4 - редактирование описания у фотографий " << endl;  cout << "5 - редактирование своих данный login/password " << endl;  cout << "6 - просмотр подписок " << endl;  cout << "7 - поиск пользователей " << endl;  cout << "8 - выход и сохранение данных " << endl;  cout << "9 - информация о программе " << endl;  cin >> choose;  switch (choose) {  case 1:  lookPhotoActiveUser();  break;  case 2:  addImage();  break;  case 3:  removePhoto();  break;  case 4:  editDescriptionPhoto();  break;  case 5:  editData();  break;  case 6:  lookFollowers();  break;  case 7:  findPeople();  break;  case 8:  saveAndClose();  break;  case 9:  info();  break;  default:  break;  }  }  }  void Instagramm::addImage(){  Photo photo;  cout << "Введите ссылку на фотографию " << endl;  cin >> photo.href;  cout << "Введите описание " << endl;  cin >> photo.description;  activeUser.photoVector.push\_back(photo);  cout << "фото успешно добавлено " << endl;  }  void Instagramm::findPeople(){  string login;  bool find = false;  cout << "Введите логин пользователя " << endl;  cin >> login;  for (int i=0;i<dataAllUser.size();i++){  if(dataAllUser[i].login == login){  find = true;  cout << "Пользователь найден " << endl;  cout << "1 - подписаться" << endl;  cout << "2 - просмотреть фотографии " << endl;  int choose;  cin >> choose;  switch (choose) {  case 1:  activeUser.followers.push\_back(dataAllUser[i].login);  cout << "Вы успешно подписались" << endl;  break;  case 2:  for (int j=0;i<dataAllUser[j].photoVector.size();j++){  cout << dataAllUser[i].photoVector[j].href << " " << dataAllUser[i].photoVector[j].description << endl;  }  if (dataAllUser[i].photoVector.size() == 0){  cout << "у пользователя нету фото" << endl;  }  break;  default:  break;  }  break;  }  }  if (!find){  cout << "Пользователь не найден " << endl;  }  }  void Instagramm::lookPhotoActiveUser(){  for (int i=0;i<activeUser.photoVector.size();i++){  cout << activeUser.photoVector[i].description << "->" << activeUser.photoVector[i].href << endl;  }  if (activeUser.photoVector.size() == 0){  cout << "у вас нету фото" << endl;  }  }  void Instagramm::lookFollowers(){  for (int i=0;i<activeUser.followers.size();i++){  cout << activeUser.followers[i] << endl;  }  if (activeUser.followers.size() == 0){  cout << "у вас нету подписчиков" << endl;  }  }  void Instagramm::editData(){  cout << "Что вы хотите изменить ?" << endl;  cout << "1 - логин" << endl;  cout << "2 - пароль" << endl;  int choose;  cin >> choose;  switch (choose) {  case 1:{  string login;  int index = 0;  cout << "Введите новый логин" << endl;  cin >> login;  for (int i=0;i<dataAllUser.size();i++){  if (dataAllUser[i].login == activeUser.login){  index = i;  break;  }  }  activeUser.login = login;  dataAllUser[index] = activeUser;  cout << "логин успешно изменен" << endl;  break;  }  case 2:{  string password,password2;  cout << "Введите новый пароль " << endl;  cin >> password;  cout << "Повторите новый пароль" << endl;  cin >> password2;  if (password == password2){  activeUser.password = password;  cout << "Пароль успешно изменен" << endl;  }else{  cout << "Пароли разные " << endl;  }  break;  }  default:  cout << "error" << endl;  break;  }  }  void Instagramm::editDescriptionPhoto(){  cout << "Выберите номер фотографии, описание которой хотите изменить" << endl;  for (int i=0;i<activeUser.photoVector.size();i++){  cout << i+1 << ":" << activeUser.photoVector[i].description << " " << activeUser.photoVector[i].href << endl;  }  int index;  cin >> index;  cout << "Вы выбрали изображение :" << endl;  cout << activeUser.photoVector[index-1].description << " " << activeUser.photoVector[index-1].href << endl;  string description;  cout << "Введите новое описание" << endl;  cin >> description;  activeUser.photoVector[index-1].description = description;  cout << "Описание успешно изменёно" << endl;  }  void Instagramm::removePhoto(){  cout << "Выберите номер фотографии, которую хотите удалить " << endl;  for (int i=0;i<activeUser.photoVector.size();i++){  cout << i+1 << ":" << activeUser.photoVector[i].description << ":" << activeUser.photoVector[i].href << endl;  }  int index;  cin >> index;  cout << "Вы выбрали изображение :" << endl;  cout << activeUser.photoVector[index-1].href << " " << activeUser.photoVector[index-1].description << endl;  activeUser.photoVector.erase(activeUser.photoVector.begin()+index-1);  cout << "Изображение успешно удалено"<< endl;  }  void Instagramm::saveAndClose(){  ofstream file(path);  if (file.is\_open()){  for (int i=0;i<dataAllUser.size();i++){  if (dataAllUser[i].login == activeUser.login){  dataAllUser[i] = activeUser;  break;  }  }  for (int i=0;i<dataAllUser.size();i++){  file << dataAllUser[i].login << "#" << dataAllUser[i].password << "#";  for (int j=0;j<dataAllUser[i].followers.size();j++){  file << dataAllUser[i].followers[j] << "!";  }  file << "\*";  for (int k=0;k<dataAllUser[i].photoVector.size();k++){  file << dataAllUser[i].photoVector[k].description << " " << dataAllUser[i].photoVector[k].href << "|";  }  file << endl;  }  }else{  cout << "файл не найден " << endl;  }  }  void Instagramm::info(){  cout << "Приложение для обмена фотографиями" << endl;  cout << "Разработчик - Кирилл Корнющенков" << endl;  cout << "Год выпуска - 2020" << endl;  } |