|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  |  | |  |
|  |  |  | |
|  |  |  | |
| // Raidziu daznis tekste  #include <fstream>  #include <iomanip>  using namespace std;  const char CDfv[] = "Duomenys11.txt"; // pradiniu duomenu failo vardas  const char CRfv[] = "Rezultatai1.txt"; // rezultatu failo vardas  const int CMax = 256; // masyvu dydis  //------------------------------------------------  int Kiek(char sim);  void Rikiuoti(char S[], int A[], int n);  //-----------------------------------------------  int main()  {  char S[CMax]; // raidziu masyvas  int A[CMax]; // raidziu pasikartojimo skaiciai  int n = 0;  for (char sim = 'a'; sim <= 'z'; sim++) {  S[n] = sim; A[n] = Kiek(sim);  n++;  }  Rikiuoti(S, A, n);  ofstream fr(CRfv);    for (int i = 0; i < n; i++) { // rasomi simboliai po penkis eiluteje  fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;  }  fr << endl;  fr.close();  return 0;  }  //---------------------------------------------  // Apskaiciuoja ir grazina simbolio sim pasikartojimo pradiniu duomenu faile skaiciu  int Kiek(char sim)  {  char s;  int k = 0;      ifstream fd(CDfv);  while (!fd.eof()) {  fd.get(s);  if (!fd.eof() && (tolower(s) == sim)) k++;  }  fd.close();  return k;  }  //---------------------------------------------  // Simboliu masyvas rikiuojamas mazejanciai pagal simboliu pasikartojimo skaiciu  // Kartu rikiuojamas ir simboliu pasikartojimo skaiciu masyvas  void Rikiuoti(char S[], int A[], int n)  {  int sk; char sim;  for (int i = 0; i < n; i++)  for (int j = i + 1; j < n; j++)  if (A[j] > A[i]) {  int sk = A[i]; A[i] = A[j]; A[j] = sk;  int sim = S[i]; S[i] = S[j]; S[j] = sim;  }  }  //----------------------------------------------- | // Raidziu daznis tekste  #include <fstream>  #include <iomanip>  using namespace std;  const char CDfv[] = "Duomenys11.txt"; // pradiniu duomenu failo vardas  const char CRfv[] = "Rezultatai1.txt"; // rezultatu failo vardas  const int CMax = 256; // masyvu dydis  //--------------------------------------------------  int Kiek(char sim);  void Rikiuoti(char S[], int A[], int n);  //-------------------------------------------------  int main()  {  char S[CMax]; // raidziu masyvas  int A[CMax]; // raidziu pasikartojimo skaiciai  int n = 0;  for (char sim = 'a'; sim <= 'z'; sim++) {  S[n] = sim; A[n] = Kiek(sim);  n++;  }  Rikiuoti(S, A, n);  ofstream fr(CRfv);  fr << "Tekste esantys simboliai:" << endl;  fr << endl;  for (int i = 0; i < n; i++) { // rasomi simboliai po penkis eiluteje  if (A[i] != 0) {fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;}  }  fr << endl;  fr << "Tekste nesantys simboliai:" << endl;  fr << endl;  for (int i = 0; i < n; i++) { // rasomi simboliai po penkis eiluteje  if (A[i] == 0) {fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;}  }  fr << endl;  fr.close();  return 0;  }  //-------------------------------------------  // Apskaiciuoja ir grazina simbolio sim pasikartojimo pradiniu duomenu faile skaiciu  int Kiek(char sim)  {  char s;  int k = 0;    ifstream fd(CDfv);  while (!fd.eof()) {  fd.get(s);  if (!fd.eof() && (tolower(s) == sim)) k++;  }  fd.close();  return k;  }  //-----------------------------------------------  // Simboliu masyvas rikiuojamas mazejanciai pagal simboliu pasikartojimo skaiciu  // Kartu rikiuojamas ir simboliu pasikartojimo skaiciu masyvas  void Rikiuoti(char S[], int A[], int n)  {  int sk; char sim;  for (int i = 0; i < n; i++)  for (int j = i + 1; j < n; j++)  if (A[j] > A[i]) {  int sk = A[i]; A[i] = A[j]; A[j] = sk;  int sim = S[i]; S[i] = S[j]; S[j] = sim;  }  }  //--------------------------------------------- | // Raidziu daznis tekste  #include <fstream>  #include <iomanip>  using namespace std;  const char CDfv[] = "Duomenys11.txt"; // pradiniu duomenu failo vardas  const char CRfv[] = "Rezultatai2.txt"; // rezultatu failo vardas  const int CMax = 256; // masyvu dydis  //-----------------------------------------  void Kiek(char S[], int A[], int & n);  void Rikiuoti(char S[], int A[], int n);  void Rasyti(const char fv[], char S[], int A[], int n);  //-----------------------------------------  int main()  {  char S[CMax]; // raidziu masyvas  int A[CMax]; // raidziu pasikartojimo skaiciai  int n = 0;  Kiek(S, A, n);    Rikiuoti(S, A, n);  Rasyti (CRfv, S, A, n);  return 0;  }  //-----------------------------------------  // Apskaiciuoja ir grazina simbolio sim pasikartojimo pradiniu duomenu faile skaiciu  void Kiek(char S[], int A[], int & n)  {  char ss;  n = 0;  for (ss = 'a'; ss <= 'z'; ss++) {  S[n] = ss; A[n] = 0; n++;  }    ifstream fd(CDfv);  while (!fd.eof()) {  fd.get(ss);  for (int i = 0; i <= n; i++)  if (!fd.eof() && (tolower(ss) == S[i])) A[i]++;  }  fd.close();  }  //--------------------------------------------  // Simboliu masyvas rikiuojamas mazejanciai pagal simboliu pasikartojimo skaiciu  // Kartu rikiuojamas ir simboliu pasikartojimo skaiciu masyvas  void Rikiuoti(char S[], int A[], int n)  {  int sk; char sim;  for (int i = 0; i < n; i++)  for (int j = i + 1; j < n; j++)  if (A[j] > A[i]) {  int sk = A[i]; A[i] = A[j]; A[j] = sk;  int sim = S[i]; S[i] = S[j]; S[j] = sim;  }  }  //----------------------------------------------  void Rasyti(const char fv[], char S[], int A[], int n) {  ofstream fr(CRfv);  fr << "Tekste esantys simboliai:" << endl;  fr << endl;  for (int i = 0; i < n; i++) { // rasomi simboliai po penkis eiluteje  if (A[i] != 0) {fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;}  }  fr << endl;  fr << "Tekste nesantys simboliai:" << endl;  fr << endl;  for (int i = 0; i < n; i++) { // rasomi simboliai po penkis eiluteje  if (A[i] == 0) {fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;}  }  fr << endl;  fr.close();  }  //----------------------------------------------- | // Raidziu daznis tekste  #include <fstream>  #include <iomanip>  using namespace std;  const char CDfv[] = "Duomenys11.txt"; // pradiniu duomenu failo vardas  const char CRfv[] = "Rezultatai3.txt"; // rezultatu failo vardas  const int CMax = 256; // masyvu dydis  //-----------------------------------------  void Kiek(char S[], int A[], int & n);  void Rikiuoti(char S[], int A[], int n);  void Rasyti(const char fv[], char S[], int A[], int n);  //-----------------------------------------  int main()  {  char S[CMax]; // raidziu masyvas  int A[CMax]; // raidziu pasikartojimo skaiciai  int n = 0;  Kiek(S, A, n);    Rikiuoti(S, A, n);  Rasyti (CRfv, S, A, n);  return 0;  }  //-----------------------------------------  // Apskaiciuoja ir grazina simbolio sim pasikartojimo pradiniu duomenu faile skaiciu  void Kiek(char S[], int A[], int & n)  {  char ss;  n = 0;  for (ss = 'a'; ss <= 'z'; ss++) {  S[n] = ss; A[n] = 0; n++;  }  S[n] = '.'; A[n] = 0; n++;  S[n] = ','; A[n] = 0; n++;  S[n] = ';'; A[n] = 0; n++;  S[n] = '-'; A[n] = 0; n++;    ifstream fd(CDfv);  while (!fd.eof()) {  fd.get(ss);  for (int i = 0; i <= n; i++)  if (!fd.eof() && (ss == S[i])) A[i]++;  }  fd.close();  }  //------------------------------------------  // Simboliu masyvas rikiuojamas mazejanciai pagal simboliu pasikartojimo skaiciu  // Kartu rikiuojamas ir simboliu pasikartojimo skaicii masyvas  void Rikiuoti(char S[], int A[], int n)  {  int sk; char sim;  for (int i = 0; i < n; i++)  for (int j = i + 1; j < n; j++)  if (A[j] > A[i]) {  int sk = A[i]; A[i] = A[j]; A[j] = sk;  int sim = S[i]; S[i] = S[j]; S[j] = sim;  }  }  //---------------------------------------------  void Rasyti(const char fv[], char S[], int A[], int n) {  ofstream fr(CRfv);  fr << "Tekste esantys simboliai:" << endl;  fr << endl;  for (int i = 0; i < n; i++) { // rasomi simboliai po penkis eiluteje  if (A[i] != 0) {fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;}  }  fr << endl;  fr << "Tekste nesantys simboliai:" << endl;  fr << endl;  for (int i = 0; i < n; i++) { // rassomi simboliai po penkis eiluteje  if (A[i] == 0) {fr << S[i] << " " << setw(2) << A[i] << " ";  if ((i + 1) % 5 == 0) fr << endl;}  }  fr << endl;  fr.close();  }  //----------------------------------------------- |