Заданне 5. В соответствии со своим вариантом разработать программы для условий, приведенных в таблице ниже, и изучить способы работы с файлами на ***языке С***.

Варыянт 3

|  |
| --- |
| 1. Дан файл вещественных чисел, содержащий элементы квадратной матрицы (по строкам), причем начальный элемент файла содержит значение количества столбцов матрицы. Создать новый файл той же структуры, содержащий транспонированную матрицу. |
| #include <iostream>  #include <cstdio>  #include <vector>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  // Адкрыццё файла для чытання  FILE\* inputFile;  // Калі не ўдаецца адкрыць файл для чытання (fopen\_s вяртае ненулявое значэнне ў выпадку памылкі)  if (fopen\_s(&inputFile, "input.txt", "r") != 0 || inputFile == nullptr) {  std::cerr << "Не атрымалася адкрыць файл для чытання." << std::endl;  return 1;  }  // Чытанне памеру матрыцы (колькасць слупкоў)  int numColumns;  if (fscanf\_s(inputFile, "%d", &numColumns) != 1) {  std::cerr << "Памылка чытання памеру матрыцы." << std::endl;  fclose(inputFile);  return 1;  }  // Стварэнне вектара для захоўвання матрыцы  std::vector<std::vector<double>> matrix;  double element;  // Чытанне матрыцы з файла  while (fscanf\_s(inputFile, "%lf", &element) == 1) { // Чытанне элемента з файла. Калі паспяхова, fscanf\_s вяртае 1 (колькасць паспяхова прачытаных элементаў)  // Калі fscanf\_s вяртае 0 (не атрымалася прачытаць элемент) ці іншае значэнне, цыкл завершыцца  std::vector<double> row; // Вектар для захоўвання радка матрыцы  row.push\_back(element); // Даданне першага элемента радка з файла ў вектар радка выкарыстоўваючы метад вектараў push\_back()  // Даданне ўсіх астатніх элементаў радка ў вектар  for (int i = 1; i < numColumns; ++i) {  if (fscanf\_s(inputFile, "%lf", &element) != 1) {  std::cerr << "Памылка чытання элемента матрыцы." << std::endl;  fclose(inputFile);  return 1;  }  row.push\_back(element);  }  // Даданне радка ў матрыцу  matrix.push\_back(row);  }  // Закрыццё файла  fclose(inputFile);  // Адкрыццё новага файла для запісу  FILE\* outputFile;  if (fopen\_s(&outputFile, "output.txt", "w") != 0 || outputFile == nullptr) {  std::cerr << "Немагчыма адкрыць файл для запісу." << std::endl;  return 1;  }  // Запіс транспанаванай матрыцы ў новы файл  for (int i = 0; i < numColumns; i++) {  for (const auto& row : matrix) {  fprintf(outputFile, "%.2lf", row[i]);  }  fprintf(outputFile, "\n");  }  // Закрыццё файла  fclose(outputFile);  return 0;  } |
| 2. Даны три файла целых чисел одинакового размера с именами **NameA**, **NameB** и **NameC**. Создать новый файл с именем **NameD**, в котором чередовались бы элементы исходных файлов с одним и тем же номером: **A0, B0, C0, A1, B1, C1, A2, B2, C2, ...** |
| #include <iostream>  #include <cstdio>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  // Адкрыццё файлаў для чытання  FILE\* fileA;  if (fopen\_s(&fileA, "NameA.txt", "r") != 0 || fileA == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameA.txt для чытання." << std::endl;  return 1;  }  FILE\* fileB;  if (fopen\_s(&fileB, "NameB.txt", "r") != 0 || fileB == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameB.txt для чытання." << std::endl;  fclose(fileA);  return 1;  }  FILE\* fileC;  if (fopen\_s(&fileC, "NameC.txt", "r") != 0 || fileC == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameC.txt для чытання." << std::endl;  fclose(fileA);  fclose(fileB);  return 1;  }  // Адкрыццё новага файла для запісу  FILE\* fileD;  if (fopen\_s(&fileD, "NameD.txt", "w") != 0 || fileD == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameD.txt для запісу." << std::endl;  fclose(fileA);  fclose(fileB);  fclose(fileC);  return 1;  }  // Чытанне і запіс дадзеных з файлаў A, B, і C у файл D  int elementA, elementB, elementC;  while (fscanf\_s(fileA, "%d", &elementA) == 1 &&  fscanf\_s(fileB, "%d", &elementB) == 1 &&  fscanf\_s(fileC, "%d", &elementC) == 1) {  // Запіс элементаў з файлаў A, B, C у файл D у паказаным парадку  fprintf(fileD, "%d %d %d ", elementA, elementB, elementC);  }  // Закрыццё файлаў  fclose(fileA);  fclose(fileB);  fclose(fileC);  fclose(fileD);  std::cout << "Аперацыя паспяхова выканана." << std::endl;  return 0;  } |

Варыянт 4

|  |
| --- |
| 1. Компоненты файла **fileA** – целые отличные от нуля положительные и отрицательные числа. Получить файл **fileB**, состоящий из положительных чисел. |
| #include <iostream>  #include <cstdio>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  // Адкрыццё файла fileA для чытання  FILE\* fileA;  if (fopen\_s(&fileA, "fileA.txt", "r") != 0 || fileA == nullptr) {  std::cerr << "Не атрымалася адкрыць файл fileA.txt для чытання." << std::endl;  return 1;  }  // Адкрыццё новага файла fileB для запісу дадатных лікаў  FILE\* fileB;  if (fopen\_s(&fileB, "fileB.txt", "w") != 0 || fileB == nullptr) {  std::cerr << "Немагчыма адкрыць файл fileB.txt для запісу." << std::endl;  fclose(fileA);  return 1;  }  int number;  // Чытанне і запіс дадатных лікаў з файла fileA у файл fileB  while (fscanf\_s(fileA, "%d", &number) == 1) {  if (number > 0) {  fprintf(fileB, "%d ", number);  }  }  // Закрыццё файлаў  fclose(fileA);  fclose(fileB);  std::cout << "Аперацыя паспяхова выканана." << std::endl;  return 0;  } |
| 2. Компоненты файла **f** – целые двухзначные числа. Получить файл **g**, образованный из **f** включением только чисел больших некоторого числа, вводимого с клавиатуры. |
| #include <iostream>  #include <cstdio>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  // Увод ліку з клавіятуры  int threshold;  std::cout << "Увядзіце парогавае значэнне: ";  std::cin >> threshold;  // Адкрыццё файла f для чытання  FILE\* fileF;  if (fopen\_s(&fileF, "fileF.txt", "r") != 0 || fileF == nullptr) {  std::cerr << "Не атрымалася адкрыць файл fileF.txt для чытання." << std::endl;  return 1;  }  // Адкрыццё новага файла g для запісу лікаў, вялікіх уведзенага значэння  FILE\* fileG;  if (fopen\_s(&fileG, "fileG.txt", "w") != 0 || fileG == nullptr) {  std::cerr << "Немагчыма адкрыць файл fileG.txt для запісу." << std::endl;  fclose(fileF);  return 1;  }  int number;  // Чытанне і запіс лікаў з файла f у файл g, калі яны больш уведзенага значэння  while (fscanf\_s(fileF, "%d", &number) == 1) {  if (number > threshold) {  fprintf(fileG, "%d ", number);  }  }  // Закрыццё файлаў  fclose(fileF);  fclose(fileG);  std::cout << "Аперацыя паспяхова выканана." << std::endl;  return 0;  } |

Варыянт 6

|  |
| --- |
| 1. Компоненты файла **fA** – вещественные числа (положительные и отрицательные). Определить и вывести на экран порядковый номер того из них, которое наиболее близко к введенному пользователем целому числу. |
| #define NOMINMAX //каб не было праблем з выкарыстаннем ::max();  #include <Windows.h>  #include <iostream>  #include <cstdio>  #include <cmath>  #include <limits>  int main() {  SetConsoleOutputCP(1251);  // Увод цэлага ліку з клавіятуры  int targetNumber;  std::cout << "Увядзіце цэлы лік: ";  std::cin >> targetNumber;  // Адкрыццё файла fA для чытання  FILE\* fileFA;  if (fopen\_s(&fileFA, "fileFA.txt", "r") != 0 || fileFA == nullptr) {  std::cerr << "Не атрымалася адкрыць файл fileFA.txt для чытання." << std::endl;  return 1;  }  double number;  double closestNumber;  int order = 0;  double minDifference = std::numeric\_limits<double>::max(); // Ініцыялізацыя зменнай minDifference максімальна магчымым значэннем тыпу double  // Лічыльнік для парадкавага нумара ліку ў файле  int counter = 0;  // Чытанне лікаў з файла fA і пошук бліжэйшага да ўведзенага ліку  while (fscanf\_s(fileFA, "%lf", &number) == 1) { // Счытванне рэчыўнага ліку з файла  double difference = std::abs(number - targetNumber); // Розніца паміж лічаным лікам і уведзеным карыстальнікам лікам  if (difference < minDifference) { // Калі розніца менш бягучай мінімальнай розніцы  closestNumber = number; // Абнаўляем бліжэйшы лік  minDifference = difference; // Абнаўляем мінімальную розніцу  order = counter + 1; // Абнаўляем парадкавы нумар ліку ў файле  }  counter++; // Павялічваем лічыльнік  }  // Закрыццё файла  fclose(fileFA);  // Выснова вынікаў  std::cout << "Бліжэйшы лік да " << targetNumber << ": " << closestNumber << std::endl;  std::cout << "Парадкавы нумар у файле: " << order << std::endl;  return 0;  } |
| 2. Создать текстовый файл **F1** не менее, чем из 6 строк, и записать в него информацию. Скопировать в файл **F2** только четные строки из **F1**. |
| #include <cstdio>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  // Стварэнне і запіс у файл F1  FILE\* fileF1;  if (fopen\_s(&fileF1, "F1.txt", "w") != 0 || fileF1 == nullptr) {  printf("Не атрымалася стварыць або адкрыць файл F1.txt для запісу.\n");  return 1;  }  // Запіс інфармацыі ў F1  fprintf(fileF1, "Радок 1\n");  fprintf(fileF1, "Радок 2\n");  fprintf(fileF1, "Радок 3\n");  fprintf(fileF1, "Радок 4\n");  fprintf(fileF1, "Радок 5\n");  fprintf(fileF1, "Радок 6\n");  // Закрыццё файла F1  fclose(fileF1);  // Адкрыццё файлаў F1 і F2 для чытання і запісы адпаведна  FILE\* fileF1Read;  FILE\* fileF2;  if (fopen\_s(&fileF1Read, "F1.txt", "r") != 0 || fileF1Read == nullptr) {  printf("Не атрымалася адкрыць файл F1.txt для чытання.\n");  return 1;  }  if (fopen\_s(&fileF2, "F2.txt", "w") != 0 || fileF2 == nullptr) {  printf("Не атрымалася стварыць або адкрыць файл F2.txt для запісу.\n");  fclose(fileF1Read);  return 1;  }  char buffer[256]; // Буфер для счытвання радка з файла  int lineNumber = 1; // Зменная для захоўвання нумара радка  // Капіяванне цотных радкоў з F1 у F2  while (fgets(buffer, sizeof(buffer), fileF1Read) != nullptr) {  if (lineNumber % 2 == 0) { // Праверка, ці з'яўляецца нумар радка цотным  fprintf(fileF2, "%s", buffer); // Запіс радка ў файл F2  }  lineNumber++; // Павелічэнне лічыльніка нумара радка  }  // Закрыццё файлаў  fclose(fileF1Read);  fclose(fileF2);  printf("Капіраванне завершана.\n");  return 0;  } |

Варыянт 7

|  |
| --- |
| 1. Даны три файла целых чисел одинакового размера с именами **NameA**, **NameB** и **NameC**. Создать новый файл с именем **NameD**, в который записать максимальные элементы исходных файлов с одним и тем же номером: **max(*A0*, *B0*, *C0)***, **max( *A1*, *B1*, *C1)***, **max(*A2*, *B2*, *C2)*, ...** |
| #include <stdio.h>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  FILE\* fileA, \* fileB, \* fileC, \* fileD;  errno\_t err;  // Адкрыццё файлаў NameA, NameB і NameC для чытання  err = fopen\_s(&fileA, "NameA.txt", "r");  if (err != 0 || fileA == NULL) {  perror("Не атрымалася адкрыць файл NameA.txt для чытання");  return 1;  }  err = fopen\_s(&fileB, "NameB.txt", "r");  if (err!= 0 || fileB == NULL) {  perror("Не атрымалася адкрыць файл NameB.txt для чытання");  fclose(fileA);  return 1;  }  err = fopen\_s(&fileC, "NameC.txt", "r");  if (err != 0 || fileC == NULL) {  perror("Не атрымалася адкрыць файл NameC.txt для чытання");  fclose(fileA);  fclose(fileB);  return 1;  }  // Адкрыццё файла NameD для запісу  err = fopen\_s(&fileD, "NameD.txt", "w");  if (err!= 0 || fileD == NULL) {  perror("Не атрымалася адкрыць файл NameD.txt для запісу");  fclose(fileA);  fclose(fileB);  fclose(fileC);  return 1;  }  int elementA, elementB, elementC;  // Чытанне і параўнанне элементаў з файлаў NameA, NameB і NameC  while (fscanf\_s(fileA, "%d", &elementA) == 1 &&  fscanf\_s(fileB, "%d", &elementB) == 1 &&  fscanf\_s(fileC, "%d", &elementC) == 1) {  int maxElement = elementA > elementB ? (elementA > elementC ? elementA : elementC) : (elementB > elementC ? elementB : elementC); // Максімальны элемент  // Запіс максімальнага элемента ў файл NameD  fprintf(fileD, "%d ", maxElement);  }  // Закрыццё файлаў  fclose(fileA);  fclose(fileB);  fclose(fileC);  fclose(fileD);  printf("Праграма паспяхова выканана.\n");  return 0;  } |
| 2. Даны три файла целых чисел одинакового размера с именами **A**, **B** и **C**. Создать новый файл с именем **D**, в котором чередовались бы элементы исходных файлов с одним и тем же номером: **a0, b0, c0, a1, b1, c1, a2, b2, c2,** ... . |
| #include <iostream>  #include <cstdio>  #include <Windows.h>  int main() {  SetConsoleOutputCP(1251);  // Адкрыццё файлаў для чытання  FILE\* fileA;  if (fopen\_s(&fileA, "NameA.txt", "r") != 0 || fileA == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameA.txt для чытання." << std::endl;  return 1;  }  FILE\* fileB;  if (fopen\_s(&fileB, "NameB.txt", "r") != 0 || fileB == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameB.txt для чытання." << std::endl;  fclose(fileA);  return 1;  }  FILE\* fileC;  if (fopen\_s(&fileC, "NameC.txt", "r") != 0 || fileC == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameC.txt для чытання." << std::endl;  fclose(fileA);  fclose(fileB);  return 1;  }  // Адкрыццё новага файла для запісу  FILE\* fileD;  if (fopen\_s(&fileD, "NameD.txt", "w") != 0 || fileD == nullptr) {  std::cerr << "Немагчыма адкрыць файл NameD.txt для запісу." << std::endl;  fclose(fileA);  fclose(fileB);  fclose(fileC);  return 1;  }  // Чытанне і запіс дадзеных з файлаў A, B, і C у файл D  int elementA, elementB, elementC;  while (fscanf\_s(fileA, "%d", &elementA) == 1 &&  fscanf\_s(fileB, "%d", &elementB) == 1 &&  fscanf\_s(fileC, "%d", &elementC) == 1) {  // Запіс элементаў з файлаў A, B, C у файл D у паказаным парадку  fprintf(fileD, "%d %d %d ", elementA, elementB, elementC);  }  // Закрыццё файлаў  fclose(fileA);  fclose(fileB);  fclose(fileC);  fclose(fileD);  std::cout << "Аперацыя паспяхова выканана." << std::endl;  return 0;  } |