**.1SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 9.3 |
| **Autor** | Andrii Godlevskyi | **Grupa** | WCY23KY1S1 |
| **Temat** | Pliki i strumienie | | |

1.Treść zadania

Napisz program, który wczytuje z klawiatury trzy nazwy plików. Dwa pierwsze pliki zawierają posortowane rekordy, następnie program łączy te pliki tak, aby zapisując je do trzeciego pliku w pliku rekordy były również posortowane oraz wyświetla na ekranie zawartość trzeciego pliku (dane każdego rekordu   
w kolejnym wierszu). Listowanie zaczyna nagłówek.

Metoda realizacji

Tworzymy funkcje porównania rekordów z pliku wejściowego do sortowania, dalej sortujemy i wynik zapisujemy do stworzonego pliku i wypisujemy na ekranie.

1. Założenia / ograniczenia dotyczące danych:

1. Dane wejściowe

nazwa\_pliku we1, nazwa\_pliku we2, nazwa\_pliku wy – wczytywana z klawiatury

1. Dane wyjściowe

Dane z pliku we1, dane z pliku we2, - wyświetlone na ekranie.

Dane z pliku wy - zapisywane do pliku i wyświetlone na ekranie.

1. Realizacja
2. Algorytm

Изображение выглядит как текст, рукописный текст, Бумажное изделие, бумага

Автоматически созданное описание



1. Kod zródłowy

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define RECORD\_SIZE 10 // Rozmiar rekordu

#define RECORDS\_PER\_LINE 10 // Ilosc rekordow w wierszu

// Deklaracja rekordu

typedef struct {

int data;

char character;

} Record;

// Funkcja do porownywania rekordow do sortowania

int compareRecords(const void \*a, const void \*b) {

Record \*recordA = (Record \*)a;

Record \*recordB = (Record \*)b;

return recordA->data - recordB->data;

}

int main() {

char filenameIn1[100], filenameIn2[100], filenameOut[100];

int n1, n2, n3, i, j, k;

// Wczytanie nazw plikow

printf("Podaj nazwe pliku wejsciowego 1 (\*.dat): ");

scanf("%s", filenameIn1);

printf("Podaj nazwe pliku wejsciowego 2 (\*.dat): ");

scanf("%s", filenameIn2);

printf("Podaj nazwe pliku wyjsciowego (\*.dat): ");

scanf("%s", filenameOut);

// Otwarcie plikow wejsciowych

FILE \*fpIn1 = fopen(filenameIn1, "rb");

if (fpIn1 == NULL) {

printf("Blad otwarcia pliku wejsciowego 1.\n");

return 1;

}

FILE \*fpIn2 = fopen(filenameIn2, "rb");

if (fpIn2 == NULL) {

printf("Blad otwarcia pliku wejsciowego 2.\n");

fclose(fpIn1);

return 1;

}

// Okreslenie liczby rekordow w każdym pliku

fseek(fpIn1, 0, SEEK\_END);

n1 = ftell(fpIn1) / sizeof(Record);

rewind(fpIn1);

fseek(fpIn2, 0, SEEK\_END);

n2 = ftell(fpIn2) / sizeof(Record);

rewind(fpIn2);

// Alokacja pamieci dla tablic rekordow

Record \*records1 = (Record \*)malloc(n1 \* sizeof(Record));

Record \*records2 = (Record \*)malloc(n2 \* sizeof(Record));

if (records1 == NULL || records2 == NULL) {

printf("Blad alokacji pamieci.\n");

fclose(fpIn1);

fclose(fpIn2);

free(records1);

free(records2);

return 1;

}

// Wczytanie rekordow z plikow wejsciowych

printf("\nDane z pliku wejsciowego 1:\n");

for (i = 0; i < n1; i++) {

fread(&records1[i], sizeof(Record), 1, fpIn1);

printf("%d %c ", records1[i].data, records1[i].character);

if ((i + 1) % RECORDS\_PER\_LINE == 0) {

printf("\n");

}

}

printf("\nDane z pliku wejsciowego 2:\n");

for (i = 0; i < n2; i++) {

fread(&records2[i], sizeof(Record), 1, fpIn2);

printf("%d %c ", records2[i].data, records2[i].character);

if ((i + 1) % RECORDS\_PER\_LINE == 0) {

printf("\n");

}

}

// Zamkniecie plikow wejsciowych

fclose(fpIn1);

fclose(fpIn2);

// Łączenie i sortowanie rekordow

n3 = n1 + n2;

Record \*records3 = (Record \*)malloc(n3 \* sizeof(Record));

if (records3 == NULL) {

printf("Blad alokacji pamieci.\n");

free(records1);

free(records2);

return 1;

}

i = 0;

j = 0;

k = 0;

while (i < n1 && j < n2) {

if (records1[i].data < records2[j].data) {

records3[k++] = records1[i++];

} else {

records3[k++] = records2[j++];

}

}

while (i < n1) {

records3[k++] = records1[i++];

}

while (j < n2) {

records3[k++] = records2[j++];

}

// Otwarcie pliku wyjsciowego

FILE \*fpOut = fopen(filenameOut, "wb");

if (fpOut == NULL) {

printf("Blad otwarcia pliku wyjsciowego.\n");

free(records1);

free(records2);

free(records3);

return 1;

}

// Zapisanie posortowanych rekordow do pliku wyjsciowego

printf("\nDane z pliku wyjsciowego:\n");

for (i = 0; i < n3; i++) {

fwrite(&records3[i], sizeof(Record), 1, fpOut);

printf("%d %c ", records3[i].data, records3[i].character);

if ((i + 1) % RECORDS\_PER\_LINE == 0) {

printf("\n");

}

}

// Zamkniecie plikow

fclose(fpOut);

free(records1);

free(records2);

free(records3);

printf("\nRekordy zapisano do pliku %s.\n", filenameOut);

return 0;

}

1. Dane wejściowe

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

1. Dane wyjściowe

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

1. Złożoność obliczeniowa algorytmu

O(1)