**.1SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 9.2 |
| **Autor** | Andrii Godlevskyi | **Grupa** | WCY23KY1S1 |
| **Temat** | Pliki i strumienie | | |

1.Treść zadania

Napisz program, który wczytuje z klawiatury dwie nazwy pliku, a następnie wczytuje z pliku binarnego (plik z punktu 1) do tablicy rekordy, po czym sortuje je wg pola int i zapisuje do kolejnego pliku listując na ekranie. 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 we, nazwa\_pliku wy – wczytywana z klawiatury

1. Dane wyjściowe

Dane z pliku we - 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 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() {

//deklaracja zmiennych

char filenameIn[100], filenameOut[100];

int n, i;

// Wczytanie nazw plikow

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

scanf("%s", filenameIn);

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

scanf("%s", filenameOut);

// Otwarcie pliku wejsciowego

FILE \*fpIn = fopen(filenameIn, "rb");

if (fpIn == NULL) {

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

return 1;

}

// Okreslenie liczby rekordow

fseek(fpIn, 0, SEEK\_END);

n = ftell(fpIn) / sizeof(Record);

rewind(fpIn);

// Alokacja pamieci dla tablicy rekordow

Record \*records = (Record \*)malloc(n \* sizeof(Record));

if (records == NULL) {

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

fclose(fpIn);

return 1;

}

// Wczytanie rekordow z pliku wejsciowego

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

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

fread(&records[i], sizeof(Record), 1, fpIn);

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

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

printf("\n");

}

}

// Zamknięcie pliku wejsciowego

fclose(fpIn);

// Sortowanie rekordow

qsort(records, n, sizeof(Record), compareRecords);

// Otwarcie pliku wyjsciowego

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

if (fpOut == NULL) {

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

free(records);

return 1;

}

// Zapisanie posortowanych rekordow do pliku wyjsciowego

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

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

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

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

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

printf("\n");

}

}

// Zamkniecie plikow

fclose(fpOut);

free(records);

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

Автоматически созданное описание

1. Złożoność obliczeniowa algorytmu

O(1)