Operacje na plikach – zapis i odczyt danych

Zasady pracy na plikach

Program wykonujący operację na plikach powinien zachować schemat działania zapewniający poprawną pracę:

- otwarcie pliku w odpowiednim trybie (do zapisu, odczytu, zapisu i odczytu),
- wykonanie operacji zapisu lub odczytu,
- zamknięcie otwartego pliku.

Otwarcie pliku

Operację otwarcia pliku wykonujemy za pomocą funkcji ***fopen***. Funkcja ta posiada dwa parametry: pierwszy - nazwę pliku, drugi - tryb otwarcia.

Jeżeli operacja otwarcia powiodła się funkcja zwraca uchwyt do pliku wykorzystywany później w operacjach zapisu, odczytu i zamknięcia pliku. Jeżeli wystąpił błąd otwarcia pliku funkcja zwraca wartość **NULL**.

Tryb otwarcia pliku podajemy w postaci łańcucha tekstowego składającego się z odpowiednich składowych. Łańcuchy tekstowe identyfikujące tryb otwarcia przedstawione zostały w poniższej tabeli.

Tekst	Opis
r	tylko do odczytu
w	tylko do zapisu. Jeżeli plik istniał wcześniej - zostanie nadpisany (poprzednie dane zostaną utracone).
a	dopisywanie do końca pliku. Jeżeli plik nie istniał wcześniej zostanie utworzony.
r+	do zapisu i odczytu dla istniejących wcześniej plików
w+	do zapisu i odczytu dla nieistniejących wcześniej plików - plik zostanie utworzony. Jeżeli plik istniał wcześniej - zostanie nadpisany (poprzednie dane zostaną utracone).
a+	dopisywanie do końca pliku. Jeżeli plik nie istniał wcześniej zostanie utworzony.

Dodatkowo możemy zdefiniować rodzaj pliku: tekstowy (**t**) lub binarny (**b**).

Przykłady

Operacja:

```
FILE *out;
out = fopen("raport.txt", "wt");
```

otworzy plik tekstowy o nazwie *raport.txt* w trybie do zapisu.

Z kolei:

```
FILE *out;
out = fopen("C:\\terefere.bin", "r+b");
```

otworzy plik binarny (ciąg bajtów) o nazwie *terefere.bin* na dysku *C:* do zapisu i odczytu istniejącego pliku - jeżeli plik nie istnieje funkcja zwróci NULL.

W celu zabezpieczenia programu przed niewłaściwym działaniem (odwołaniem do zerowego uchwytu) należy sprawdzić poprawność otwarcia pliku, np.:

```
FILE *out;
if ((out = fopen("raport.txt", "wt"))== NULL)
{
    printf("\nBlad otwarcia pliku!\n");
    return 1;
}
```

W tej instrukcji korzystamy z operacji działających na plikach z pakietu standardowych operacji wejścia/wyjścia (plik nagłówkowy *stdio.h*) - *fopen*, *fprintf*, *fscanf*, *fread*, *fwrite*, *fclose* itd.

Operacje zapisu lub odczytu

W zależności od rodzaju danych (tekstowe, binarne) używamy funkcji:

- do zapisu danych tekstowych:
  - funkcja **fprintf**, np.:

    ```
    fprintf(out, "\n To jest tekst zapisywany do pliku \n");
    ```

    gdzie: *out* - uchwyt do pliku otwartego do zapisu. Działanie identyczne jak znanej funkcji *printf*. Dodatkowo dochodzi pierwszy parametr (właśnie *out*).
  - funkcja **fputs**, np.:

    ```
    fputs("\n To jest tekst zapisywany do pliku \n", out);
    ```

    gdzie: *out* - uchwyt do pliku otwartego do zapisu.
- do odczytu danych tekstowych:
  - funkcja **fscanf**, np.:

    ```
    fscanf(in, "%s", bufor);
    ```

    gdzie: *in* - uchwyt do pliku otwartego do odczytu; *bufor* - tablica znakowa; Funkcja odczyta jedno słowo tekstu z pliku do bufora - tablicy. Takie wywołanie czyta tekst do momentu napotkania znaku spacji lub końca linii (*eol*). Działanie identyczne jak znanej funkcji *scanf*. Dodatkowo dochodzi pierwszy parametr (*in*).
  - funkcja **fgets**, np.:

    ```
    fgets(bufor, 512, in);
    ```

    gdzie: *in* - uchwyt do pliku otwartego do odczytu; *bufor* - tablica znakowa; Funkcja odczyta jedną linię tekstu (do napotkania *end-of-line*) z pliku nie dłuższą niż 512 bajtów (wielkość bufora) do bufora - tablicy.
- do zapisu danych binarnych:
  - funkcja **fwrite**, np.:

    ```
    struct osoba {
    char imie[25];
    char nazwisko[25];
    int wiek;
    };
    ...
    osoba student;
    ...
    fwrite(&student, sizeof(osoba), 1, plik);
    ```

    gdzie: *plik* - uchwyt do pliku otwartego do zapisu; Funkcja zapisuje do pliku (uchwyt *plik*) jeden rekord o wielkości wyznaczonej przez *sizeof(osoba)* ze zmiennej *student* typu *osoba*. Pierwszy parametr to wskaźnik na bufor z którego dane zapisujemy do pliku.
- do odczytu danych binarnych:
  - funkcja **fread**, np.:

```
// definicja struktury osoba
typedef struct {
char imie[25];
char nazwisko[25];
int wiek;
} osoba;
...
osoba student;
...
fread(&student, sizeof(osoba), 1, plik);
```

gdzie: *plik* - uchwyt do pliku otwartego do odczytu; Funkcja czyta z pliku (uchwyt *plik*) jeden rekord o wielkości wyznaczonej przez *sizeof(osoba)* do zmiennej *student* typu *osoba*. Pierwszy parametr to wskaźnik na bufor do którego dane pobieramy z pliku.

UWAGA!!!

Funkcje wymienione powyżej to jedynie kilka najczęściej używanych operacji na pliku. Istnieje jeszcze kilkanaście innych.

Zamknięcie otwartego pliku

Zawsze po zakończeniu pracy na pliku należy zamknąć otwarty plik, np.:

```
fclose(out);
```

gdzie: *out* - uchwyt do pliku otwartego w dowolnym trybie;

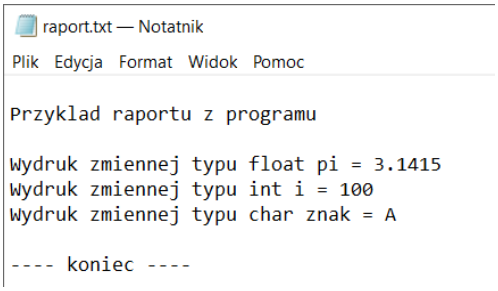
Przykłady

1. Zapis do pliku tekstowego *raport.txt*

```
#include <stdio.h>
int main()
{
    FILE *out;
    float pi = 3.1415;
    int i = 100;
    char znak = 'A';
    // otwarcie pliku tekstowego do zapisu
    if ((out = fopen("raport.txt", "wt"))== NULL)
    {
        printf("Nie mozna otworzyc pliku!\n");
        return 1;
    }
    // fprintf - zapis do pliku out
    fprintf(out,"\nPrzyklad raportu z programu\n\n");
    fprintf(out,"Wydruk zmiennej typu float pi = %6.4f \n",pi);
    fprintf(out,"Wydruk zmiennej typu int i = %3i \n",i);
    fprintf(out,"Wydruk zmiennej typu char znak = %c \n",znak);
    fprintf(out,"\n---- koniec ----\n\n");
    // zamkniecie pliku
    fclose(out);
    printf("\nDane zostaly zapisane do pliku raport.txt");
    getchar();
    return 0;
}
```

```
Dane zostaly zapisane do pliku raport.txt

--------------------------------
Process exited after 1.884 seconds with return value 0
Press any key to continue . . .
```



2. Odczyt z pliku tekstowego *raport.txt*

```c
#include <stdio.h>
int main()
{
    FILE *in;
    char bufor[512];
    // otwarcie pliku tekstowego do odczytu: rt
    if ((in = fopen("raport.txt", "rt"))== NULL)
    {
        printf("\nBlad otwarcia pliku!\n");
        return 1;
    }
    printf("\n--------- zawartosc pliku raport.txt-------------\n");
    // odczyt z pliku
    while (feof(in)==0) // feof - end of file (koniec pliku)
    {
        fgets(bufor, 512, in); // odczyt jednej linii tekstu
        printf("%s",bufor);
    }
    printf("\n------------------------------------------------\n");
    // zamkniecie pliku
    fclose(in);
    getchar();
    return 0;
}
```

```
--------- zawartosc pliku raport.txt-------------

Przyklad raportu z programu

Wydruk zmiennej typu float pi = 3.1415
Wydruk zmiennej typu int i = 100
Wydruk zmiennej typu char znak = A

---- koniec ----



------------------------------------------------


--------------------------------
Process exited after 1.607 seconds with return value 0
Press any key to continue . . .
```

3. Zapis do pliku binarnego *records.dat*

```c
#include <stdio.h>
#include <string.h> // memset
// definicja struktury osoba
typedef struct {
    char imie[25];
    char nazwisko[25];
    int wiek;
    } osoba;
int main()
{
    FILE *plik;
    int n, i;
    osoba ludzie[100];
    memset (ludzie, 0, sizeof(ludzie)); // wyczyszczenie tablicy ludzie
    printf("Zapisywanie rekordow do pliku\n");
    printf("Ile rekordow chcesz zapisac ? :\n");
    scanf("%i",&n);
    // otwarcie pliku do zapisu - handler: uchwyt pliku
    if ((plik = fopen("records.dat", "wb")) == NULL)
    {
        printf("Blad otwarcia pliku!\n");
        return 1;
    }
    fseek(plik, 0L, SEEK_SET); // ustawienie pozycji w pliku na poczatek
    for (i = 1; i<=n; i++)
    {
        // gromadzenie danych o rekordzie (osobie)
        printf("\n Imie:");
        scanf("%s",ludzie[i].imie);
        printf(" Nazwisko:");
        scanf("%s",ludzie[i].nazwisko);
        printf(" Wiek:");
        scanf("%i",&ludzie[i].wiek);
        // zapis jednego rekordu do pliku
        fwrite(&ludzie[i], sizeof(osoba), 1, plik);
    }
    fclose(plik); // zamkniecie pliku
    getchar();
    return 0;
}
```

```
Zapisywanie rekordow do pliku
Ile rekordow chcesz zapisac ? :
2

 Imie:Ola
 Nazwisko:Olowska
 Wiek:22

 Imie:Ala
 Nazwisko:Alowska
 Wiek:33

--------------------------------
Process exited after 28.5 seconds with return value 0
Press any key to continue . . .
```

4. Odczyt z pliku binarnego *records.dat*

```c
#include <stdio.h>
#include <string.h>
// definicja struktury osoba
typedef struct {
    char imie[25];
    char nazwisko[25];
    int wiek;
    } osoba;
int main()
{
    FILE *plik;
    int n, licznik=-1;
    osoba ludzie[100];
    printf("\n\rProgram odczytuje liste rekordow z pliku records.dat\n\r");
    memset (ludzie, 0, sizeof(ludzie)); // wyczyszczenie tablicy ludzie
    // otwarcie pliku do odczytu - plik: uchwyt pliku
    if ((plik = fopen("records.dat", "rb")) == NULL)
    {
        printf("Blad otwarcia pliku!\n");
        return 1;
    }
    // ustawienie pozycji w pliku na poczatek
    fseek(plik, 0L, SEEK_SET);
    // odczyt kolejnych rekordow
    while(1) // petla nieskonczona (wyjscie poprzez break)
    {
        licznik++;
        // odczyt jednego rekordu
        fread(&ludzie[licznik], sizeof(osoba), 1, plik);
        if (feof(plik)) break; // wyjscie z petli jezeli koniec pliku
        printf("\n\r Nr rekordu:%i\n\r",licznik);
        printf(" Imie: %s\n\r", ludzie[licznik].imie);
        printf(" Nazwisko: %s\n\r", ludzie[licznik].nazwisko);
        printf(" Wiek: %i\n\r", ludzie[licznik].wiek);
    }
    printf("\n\rOdczytano %i rekord(y)ow\n\r", licznik);
    fclose(plik);
    getchar();
    return 0;
}
```

```
Program odczytuje liste rekordow z pliku records.dat

 Nr rekordu:0
 Imie: Ola
 Nazwisko: Olowska
 Wiek: 22

 Nr rekordu:1
 Imie: Ala
 Nazwisko: Alowska
 Wiek: 33

Odczytano 2 rekord(y)ow


--------------------------------
Process exited after 1.668 seconds with return value 0
Press any key to continue . . .
```

UWAGA! W przykładach 3 i 4 wykorzystane zostały struktury, o których więcej w osobnej instrukcji poświęconej strukturom i uniom

Zadania

1. Przepisz i przeanalizuj działanie 4 powyższych przykładów obrazujących operacje zapisu i odczytu danych.
2. Dany jest program będący zapętlonym kalkulatorem obsługującym 4 podstawowe działania: dodawanie (+), odejmowanie (-), mnożenie (*), dzielenie(/) – plik *zad2IO.c*. Dopisz do tego kodu linie umożliwiające zapis do pliku tekstowego całej historii działania programu, tzn.

wszystkich informacji wyświetlanych na konsoli przez program i danych wprowadzanych przez użytkownika podczas działania programu.