Laboratorium 9: „Pliki”

dr inż. Arkadiusz Chrobot
dr inż. Grzegorz Łukawski

4 września 2026

1. Wprowadzenie

W niniejszej instrukcji zawarto informacje na temat obsługi plików w języku C za pomocą strumieni.

2. Pliki

Język C oferuje dwa sposoby dostępu do plików. Pierwszy nazywany jest *niskopoziomowym* i korzysta z tzw. deskryptorów plików, a drugi nazywany *wysokopoziomowym*, pozwala na dostęp do pliku za pomocą tzw. strumieni. W tej instrukcji zostanie opisany sposób wysokopoziomowy. Strumień może być potraktowany jako wskaźnik do zawartości pliku[1]. Typ takiego wskaźnika określony jest w języku C jako `FILE *`.

Do otwarcia pliku używana jest funkcja `fopen()`. Przyjmuje ona dwa argumenty. Pierwszy jest ścieżką do pliku, który ma być otwarty, zapisaną w postaci łańcucha znaków, a drugi łańcuchem znaków określającym sposób dostępu do zawartości pliku. Tabela 1 zawiera opis wartości jakie może przyjmować ten łańcuch (mogą one być ze sobą łączone, jeśli są zgodne). Domyślnie wszystkie pliki są traktowane

Tryb	Opis
`r`	Otwarcie tylko do odczytu. Wskaźnik pliku wskazuje na jego początek.
`r+`	Otwarcie do odczytu i zapisu. Wskaźnik pliku wskazuje na jego początek.
`w`	Utworzenie pliku i otwarcie tylko do zapisu. Jeśli plik o podanej nazwie istnieje, to jego zawartość jest kasowana. Wskaźnik pliku wskazuje na jego początek.
`w+`	Jak wyżej, ale z możliwością odczytu.
`a`	Otwarcie do dopisywania. Wskaźnik pliku wskazuje na jego koniec. Jeśli plik nie istnieje, to będzie utworzony.
`a+`	Jak wyżej, ale z możliwością odczytu.

Tabela 1: Tryby dostępu do zawartości pliku dla `fopen()`

przez `fopen()` jako pliki tekstowe. We wcześniejszych wersjach standardu języka C istniała możliwość otwarcia pliku jako pliku binarnego. Obecnie można do łańcucha znaków określających tryb dostępu dodać litrę „b", ale jest ona w większości przypadków ignorowana przez tę funkcję. Jeśli otwarcie pliku się powiedzie, to funkcja `fopen()` zwróci wskaźnik do niego (strumień), w przeciwnym przypadku zwróci wartość `NULL`. Plik zamyka funkcja `fclose()`. Należy zawsze pamiętać o jej wywołaniu przed zakończeniem programu. Inaczej, jeśli dokonywaliśmy zmian w zawartości pliku, to mogą one zostać utracone. Funkcja ta przyjmuje jeden argument wywołania, jakim jest strumień do pliku i zwraca zero, jeśli się wykona prawidłowo, a w przeciwnym razie wartość określoną stałą `EOF`.

Do odczytu pojedynczych znaków jest używana funkcja `fgetc()`, która zwraca wartość typu `int`, a pobiera jako argument strumień do pliku. Zapisu pojedynczego znaku można dokonać wywołując funkcję `fputc`, która jako argumenty wywołania pobiera znak zapisany w postaci zmiennej typu `int` oraz strumień. Zwraca ona wartość typu `int`, która, jeśli jest równa wartości stałej `EOF`, to sygnalizuje błąd wykonania operacji.

Do odczytu całego wiersza znaków możemy użyć funkcji `fgets()`, która przyjmuje trzy argumenty wywołania: tablicę, gdzie ma być zapisany odczytywany wiersz, liczbę elementów tej tablicy oraz strumień. Zwraca ona adres tablicy do której został zapisany wiersz (czyli swój pierwszy argument) lub `NULL` w przypadku wystąpienia błędu. Do zapisu pojedynczego wiersza używana jest funkcja `fputs()`. Przyjmuje dwa argumenty wywołania: tablicę zawierającą tekst wiersza do zapisu oraz strumień. Zwraca wartość typu `int`, która, jeśli jest równa `EOF`, to sygnalizuje wystąpienie błędu.

Aby odczytać informacje z pliku, które są różnych typów możemy zastosować funkcję `fscanf()`, której wywołanie różni się tym od wywołania funkcji `scanf()`, że jako pierwszy argument pobiera strumień. Zapis takich wartości do pliku może być dokonany za pomocą funkcji `fprintf()`, która w użyciu różni się tym od funkcji `printf()`, że również przyjmuje jako pierwszy argument strumień do pliku.

Funkcje `fread()` i `fwrite()` pozwalają odpowiednio odczytać i zapisać informację w postaci ciągu bajtów nie zakończonych znakiem (znakami) końca wiersza, czyli w postaci binarnej. Obie przyjmują

[1] Faktyczna budowa strumienia jest trochę bardziej skomplikowana.

cztery argumenty wywołania. Pierwszym jest bufor (zmienna typu `void *`) na odczytane dane (`fread()`) lub zawierająca dane do zapisu (`fwrite()`). Drugi argument to rozmiar pojedynczej porcji danych, wyrażony w bajtach. Trzeci argument to liczba porcji danych, które mają być odczytane/zapisane. Ostatni argument wywołania tych funkcji to strumień do pliku. Funkcje zwracają wartość typu `int`, która jest liczbą odczytanych/zapisanych porcji danych. Są to jedyne funkcje w języku C dotyczące obsługi plików za pomocą strumieni, które traktują plik jako plik binarny.

Każda z wymienionych funkcji zapisu lub odczytu danych zmienia położenie wskaźnika pliku przesuwając go „do przodu". Funkcja `fseek()` pozwala go modyfikować w dowolny, dozwolony sposób. Przyjmuje ona trzy argumenty wywołania. Pierwszy jest strumieniem do pliku, drugi liczbą bajtów, o jaką ma być przesunięty wskaźnik pliku, a trzeci punktem od którego należy zacząć operację przesuwania wskaźnika. Ten argument przyjmuje trzy wartości określone następującymi stałymi: `SEEK_SET` - od początku pliku, `SEEK_CUR` - względem bieżącego położenia wskaźnika i `SEEK_END` - względem końca pliku. Funkcja `ftell()` zwraca bieżące położenie wskaźnika pliku. Przyjmuje jeden argument jakim jest strumień. Obie funkcje zwracają `-1`, jeśli wykonywana przez nie operacja nie powiedzie się lub nie jest możliwa do przeprowadzenia.

W konstruowaniu pętli odczytu pliku pomocna jest funkcja `feof()`, która zwraca wartość różną od zera, jeśli został osiągnięty znacznik końca pliku. Do kontroli, czy nie wystąpił błąd operacji na pliku używana jest funkcja `ferror()`. Zwraca ona wartość niezerową, jeśli ustawiony jest znacznik błędu dla strumienia. Obie funkcje jako argument wywołania pobierają strumień. Listing 1 zawiera program, w którym zademonstrowano użycie niektórych z opisanych wyżej funkcji.

3. Zadania

Uwaga! Wszystkie programy należy napisać z podziałem na funkcje z parametrami i zaimplementować w nich obsługę wyjątków związanych z plikami. *Pliki tekstowe o stosunkowo dużych rozmiarach można znaleźć na stronie Projektu Gutenberg:* `https://www.gutenberg.org/`.

1. Napisz program, który odczyta i wypisze na ekran zawartość dowolnego pliku tekstowego, którego nazwa zostanie mu przekazana jako argument wywołania.

2. Napisz program, który odczyta swój kod źródłowy i wypisze na ekranie wyłącznie umieszczone w nim komentarze.

3. Napisz program, który zapisze do pliku macierz o wymiarach 4×4, a następnie ją odczyta i wypisze na ekran. Wartości elementów macierz powinny być losowane z przedziału $(-10, 10)$. Zarówno w pliku, jak i na ekranie macierz ma być podzielona na wiersze.

4. Napisz program, który porówna ze sobą dwa pliki tekstowe, których nazwy zostaną przekazane mu jako argumenty wywołania. Porównanie to należy zrealizować wiersz po wierszu, wypisując na ekranie te wiersze, które się różnią treścią, wraz z ich numerem i nazwą pliku z jakiego pochodzą. Numer powinien być nadawany wierszowi względem początku pliku, czyli pierwszy wiersz powinien mieć numer jeden, drugi dwa, itd.

5. Napisz program, który będzie pozwalał zapisać, dopisać i odczytać z pliku takie dane osób jak: imię, nazwisko, wiek, wzrost i waga. Program powinien umożliwić użytkownikowi dodanie dowolnej liczby zestawów danych o osobach.

6. Napisz program, który zapisze do pliku 20 liczb typu `int` wylosowanych z zakresu od -10 do 10, a następnie wypisze je na ekran i poda, która z nich jest największa, a która najmniejsza.

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

void zapis(FILE *f)
{
        int i;
        srand(time(NULL));
        for(i=0;i<10;i++)
                fprintf(f,"%d\n",1+rand()%10);
}

void odczyt(FILE *f)
{
        int a;
        while(!feof(f)) {
                fscanf(f,"%d",&a);
                if(!feof(f))
                        printf("%d\n",a);
        }

}

int main(void)
{
        FILE *plik;
        plik = fopen("test.txt","w+");
        if(plik == NULL) {
                printf("Błąd otwarcia pliku!\n");
                return 0;
        }
        zapis(plik);
        if(fseek(plik,0,SEEK_SET)==-1) {
                printf("Błąd operacji przesuwania wskaźnika pliku!\n");
                return 0;
        }
        odczyt(plik);
        if(fclose(plik))
                printf("Błąd zamknięcia pliku!\n");
        return 0;
}
```

Listing 1: Operacje na pliku