Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja
do pracowni specjalistycznej z przedmiotu

**Informatyka 1**

Kod przedmiotu: **EMS1A1004**

(studia stacjonarne)

# JĘZYK C - PLIKI BINARNE

Numer ćwiczenia

**INF1_E13**

Autor:
dr inż. Jarosław Forenc

Białystok 2025

# Spis treści

**1. Opis stanowiska ..... 3**

1.1. Stosowana aparatura ..... 3

1.2. Oprogramowanie ..... 3

**2. Wiadomości teoretyczne ..... 3**

2.1. Pliki binarne ..... 3

2.2. Operacje na plikach binarnych ..... 4

**3. Przebieg ćwiczenia ..... 9**

**4. Literatura ..... 11**

**5. Pytania kontrolne ..... 12**

**6. Wymagania BHP ..... 12**

Materiały przygotowane w ramach projektu „PB 5.0 - dostosowanie oferty dydaktycznej Politechniki Białostockiej do potrzeb nowoczesnej gospodarki oraz zielonej i cyfrowej transformacji" (nr umowy FERS.01.05-IP.08-0327/23-00) w ramach programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027 współfinansowanego ze środków Europejskiego Funduszu Społecznego Plus.

# 1. Opis stanowiska

## 1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10/11.

## 1.2. Oprogramowanie

Na komputerach zainstalowany jest edytor kodu źródłowego Visual Studio Code 1.103 (lub nowszy) wraz z odpowiednimi rozszerzeniami (C/C++, Code Runner, Polish Language Pack for Visual Studio Code) oraz MinGW - zestaw kompilatorów różnych języków programowania (m.in. C, C++, Fortran, Java).

# 2. Wiadomości teoretyczne

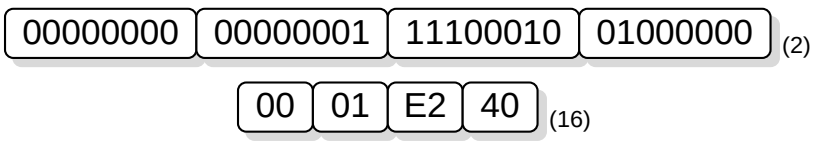
## 2.1. Pliki binarne

Plik binarny, w przeciwieństwie do pliku tekstowego, nie ma ściśle określonej struktury. Jeśli dane w pliku są przechowywane w sposób zgodny z ich reprezentacją w pamięci komputera, to mówimy, że są one zapisane w postaci binarnej.

Załóżmy, że w programie została zadeklarowana i zainicjalizowana zmienna **x** typu **int**:

```
int x = 123456;
```

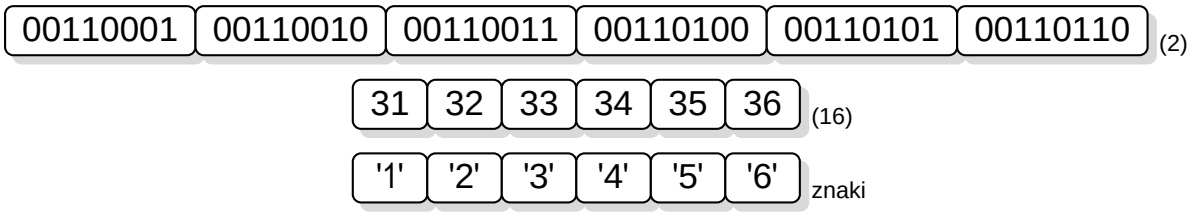
Liczba ta zajmuje w pamięci komputera 4 bajty (Rys. 1).




Rys. 1. Sposób przechowywania liczby całkowitej 123456 (typ int) w pamięci komputera (zapis w systemie dwójkowym i szesnastkowym)

Sposób przechowywania liczby **x** w pliku binarnym jest taki sam, jak w pamięci komputera (Rys. 1).

W przypadku zapisania liczby **x** do pliku tekstowego, znajdzie się w nim 6 bajtów zawierających kody ASCII kolejnych cyfr tej liczby: **'1'**, **'2'**, **'3'**, **'4'**, **'5'**, **'6'** (Rys. 2).




Rys. 2. Sposób przechowywania liczby całkowitej 123456 (typ **int**) w pliku tekstowym (zapis w systemie dwójkowym i szesnastkowym)

## 2.2. Operacje na plikach binarnych

Na plikach binarnych wykonywane są dwie podstawowe operacje:

- zapis do pliku - funkcja **fwrite()**;
- odczyt z pliku - funkcja **fread()**.

| `fwrite()` | Nagłówek: **size_t fwrite(const void *p, size_t s, size_t n, FILE *stream);** |
|---|---|

- funkcja **fwrite()** zapisuje **n** elementów o rozmiarze **s** bajtów każdy, do pliku wskazywanego przez **stream**, biorąc dane z obszaru pamięci wskazywanego przez **p**;
- funkcja zwraca liczbę zapisanych elementów - jeśli jest ona różna od **n**, to wystąpił błąd zapisu (brak miejsca na dysku lub dysk zabezpieczony przed zapisem).

Poniższy przykład zawiera deklaracje i inicjalizacje zmiennych różnych typów oraz sposób ich zapisu do pliku binarnego, wskazywanego przez zmienną o nazwie **fp**.

```
int   x = 10;
int   tab[5] = {1,2,3,4,5};
float y = 1.2345;
FILE *fp;

fp = fopen("dane.dat","wb");

fwrite(&x,sizeof(int),1,fp);
fwrite(tab,sizeof(int),5,fp);
fwrite(tab,sizeof(tab),1,fp);
fwrite(&y,sizeof(float),1,fp);
```

Pierwszym argumentem funkcji **fwrite()** jest adres w pamięci komputera, spod którego czytane są dane do zapisania w pliku. Z tego względu przed zmiennymi **x** i **y** pojawia się znak **&**. Nazwa tablicy **tab** jest adresem zerowego jej elementu więc znak **&** nie jest potrzebny. Drugi argument funkcji **fwrite()** określa rozmiar jednego elementu zapisywanego do pliku. Do określenia rozmiaru we wszystkich przypadkach zastosowano operator **sizeof**. Trzecim argumentem jest liczba zapisywanych elementów. Tablica **tab** zapisywana jest dwukrotnie. Za pierwszym razem zapisywanych jest 5 elementów tablicy, każdy o rozmiarze 4 bajty. Za drugim razem zapisywana jest od razu cała tablica (drugi argument funkcji **fwrite()** jest równy rozmiarowi całej tablicy, zaś trzeci jest równy **1**). W obu przypadkach całkowita liczba zapisywanych bajtów jest taka sama (**20**). Ostatni argument funkcji **fwrite()** wskazuje plik, do którego mają być zapisane dane.

W kolejnym programie do pliku binarnego **liczby.dat** zapisywanych jest 10 wygenerowanych pseudolosowo liczb całkowitych (typ **int**). Dla uproszczenia zapisu pominięto sprawdzenie poprawności otwarcia pliku.

Zapisanie 10 liczb całkowitych do pliku binarnego.

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

int main(void)
{
```

```
    FILE  *fp;
    int   x;

    srand((unsigned int)time(NULL));

    fp = fopen("liczby.dat","wb");
    for (int i=0; i<10; i++)
    {
        x = rand() % 100;
        fwrite(&x,sizeof(int),1,fp);
    }
    fclose(fp);

    return 0;
}
```

| `fread()` | Nagłówek: **size_t fread(void *p, size_t s, size_t n, FILE *stream);** |
|---|---|

- funkcja **fread()** pobiera **n** elementów o rozmiarze **s** bajtów każdy, z pliku wskazywanego przez **stream** i umieszcza odczytane dane w obszarze pamięci wskazywanym przez **p**;
- funkcja zwraca liczbę odczytanych elementów - w przypadku gdy liczba ta jest różna od **n**, to wystąpił błąd końca strumienia (w pliku było mniej elementów niż podana wartość argumentu **n**).

Przy założeniu, że nie jest znana ilość liczb w pliku **liczby.dat**, odczytanie pliku i wyświetlenie liczb na ekranie będzie wyglądało następująco.

Odczytanie liczb całkowitych z pliku binarnego.

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int x, ile = 0;
```

```
    fp = fopen("liczby.dat","rb");

    fread(&x,sizeof(int),1,fp);
    while (feof(fp) == 0)
    {
        printf("%d\n",x);
        ile++;
        fread(&x,sizeof(int),1,fp);
    }
    fclose(fp);
    printf("Odczytano: %d liczb\n",ile);

    return 0;
}
```

Odczytanie pierwszej liczby (wywołanie funkcji **fread()**) następuje przed pętlą **while**. Jeśli podczas tej operacji nie osiągnięto końca pliku, to funkcja **feof()** zwraca wartość równą zero i następuje wejście do pętli. W pętli wyświetlana jest na ekranie odczytana liczba **x** i zwiększana wartości zmiennej **ile** o jeden (**ile++**). Na koniec pętli odczytywana jest kolejna liczba z pliku. Warunek w pętli **while** można zapisać także w uproszczonej postaci:

```
while (!feof(fp))
```

W kolejnym przykładzie do pliku binarnego **dane.dat** zapisywana jest jedna struktura przechowująca dane osobowe oraz tablica przechowująca 5 liczb typu **float**.

Zapisanie struktury i tablicy do pliku binarnego.

```
#include <stdio.h>

struct osoba
{
    char imie[15];
    char nazw[20];
    int wiek;
};
```

```c
int main(void)
{
    FILE *fp;
    struct osoba os = {"Jan","Nowak",23};
    float tab[5] = {3.5f, 2.1f, -3.7f, 0.0f, 8.2f};

    fp = fopen("dane.dat","wb");

    fwrite(&os,sizeof(struct osoba),1,fp);
    fwrite(tab,sizeof(float),5,fp);

    fclose(fp);

    return 0;
}
```

W przypadku sprawdzania rozmiaru struktury należy po operatorze **sizeof** zawsze podawać nazwę typu strukturalnego (**struct osoba**) lub nazwę zmiennej strukturalnej (**os**). Nie można natomiast określać rozmiaru struktury na podstawie sumy rozmiarów jej pól. Jest to spowodowane tym, że kompilatory mogą pomiędzy polami struktury lub na jej końcu wstawiać dodatkowe bajty (tzw. wypełniacze). Dzięki temu pola struktury będą rozpoczynały się od adresów podzielnych przez 4. W powyższym przykładzie suma rozmiarów pól struktury wynosi **39** (15 × 1 bajt + 20 × 1 bajt + 1 × 4 bajty = 39 bajtów), zaś operator **sizeof** zwraca dla całej struktury wartość **40**.

Oprócz wymienionych funkcji **fwrite()** i **fread()**, do operacji na plikach binarnych stosowane są również funkcje opisane poniżej.

| `fseek()` | Nagłówek: int **fseek(FILE *stream,** long int **offset,** int **mode);** |
|---|---|

- pozwala przejść bezpośrednio do dowolnego bajtu w pliku wskazywanym przez **stream**;
- **offset** określa wielkość przejścia w bajtach, zaś **mode** - punkt początkowy, względem którego określane jest przejście (**SEEK_SET** - początek pliku, **SEEK_CUR** - bieżąca pozycja, **SEEK_END** - koniec pliku);

- gdy wywołanie jest poprawne, to funkcja zwraca wartość **0**; gdy wystąpił błąd (np. próba przekroczenia granic pliku), to funkcja zwraca wartość **-1**.

| `ftell()` | Nagłówek: long int **ftell(FILE *stream);** |
|---|---|

- zwraca bieżące położeniu w pliku wskazywanym przez **stream** (liczbę bajtów od początku pliku).

| `fgetpos()` | Nagłówek: int **fgetpos(FILE *stream, fpos_t *pos);** |
|---|---|

- zapamiętuję pod zmienną **pos** bieżące położenie w pliku wskazywanym przez **stream**;
- zwraca **0**, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd.

| `fsetpos()` | Nagłówek: int **fsetpos(FILE *stream,** const **fpos_t *pos);** |
|---|---|

- przechodzi do położenia **pos** w pliku wskazywanym przez **stream**;
- zwraca **0**, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd.

| `rewind()` | Nagłówek: void **rewind(FILE *stream);** |
|---|---|

- ustawia wskaźnik pozycji w pliku wskazywanym przez **stream** na początek pliku.

# 3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Plik binarny **power.dat** zawiera wyniki pomiarów mocy czynnej w postaci liczb rzeczywistych pojedynczej precyzji. Napisz program, który wyświetli na ekranie liczby odczytane z tego pliku oraz obliczy i wyświetli ich ilość, sumę i średnią arytmetyczną. Plik **power.dat** wskaże prowadzący zajęcia.

2. W pliku binarnym **pomiar.dat** zapisane są naprzemiennie wyniki pomiarów wartości chwilowych napięcia **u(t)** i prądu **i(t)** (liczby zmiennoprzecinkowe pojedynczej precyzji). Napisz program, który odczyta zawartość pliku **pomiar.dat** i na jego podstawie utworzy plik tekstowy **moc.txt** zawierający trzy kolumny liczb oznaczające: wartość chwilową napięcia, wartość chwilową prądu, wartość chwilową mocy. Plik **pomiar.dat** wskaże prowadzący zajęcia.

| u(t) | i(t) | u(t) | i(t) | u(t) | i(t) | … |
|---|---|---|---|---|---|---|

3. Plik binarny **hdd.dat** zawiera struktury **dysk** opisujące dysk twardy. Kolejne pola struktury opisują: **producenta**, **model**, **pojemność** (w GB), **prędkość obrotową** (w obr/min), **cenę** (w PLN). Plik **hdd.dat** wskaże prowadzący zajęcia.

```
struct dysk
{
    char producent[20];
    char model[20];
    int pojemnosc;
    int predkosc;
    int cena;
};
```

   Napisz program, który:

   a) odczyta zawartość pliku i wyświetli dane o dyskach na ekranie;
   b) zapisze odczytane dane do pliku tekstowego **hdd.txt** (jeden wiersz pliku powinien zawierać dane jednego dysku);
   c) obliczy i wyświetli średnią cenę dysków o pojemności większej lub równej 1000 GB i średnią cenę dysków o pojemności mniejszej od 1000 GB.

4. W pliku binarnym **pressure.dat** zapisane są wyniki pomiarów ciśnienia pochodzące z trzech czujników. Dla każdego pomiaru zapisany jest **numer czujnika** (liczba całkowita typu **int** o wartości **1**, **2** lub **3**) oraz wartość **ciśnienia w hektopaskalach** (liczba rzeczywista pojedynczej precyzji). Struktura pliku jest następująca:

| | | | | | | |
|---|---|---|---|---|---|---|
| nr | ciśnienie | nr | ciśnienie | nr | ciśnienie | … |

Napisz program, który odczyta dane z pliku binarnego i dla każdego pomiaru wyświetli na ekranie: **numer czujnika**, **wartość ciśnienia w hektopaskalach [hPa]** (z dokładnością do dwóch cyfr po kropce dziesiętnej), **wartość ciśnienia w milimetrach słupa rtęci [mmHg]** (z dokładnością do dwóch cyfr po kropce dziesiętnej). Uwagi: **1 hPa = 0,750062 mmHg**.

# 4. Literatura


[1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[3] Deitel P.J., Deitel H.: Język C. Solidna wiedza w praktyce. Wydanie VIII. Helion, Gliwice, 2020.

[4] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.

[5] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.

[6] Stańczyk J.: Nowoczesny C: przegląd C23 z przykładami. Helion, Gliwice, 2023.

[7] http://www.cplusplus.com/reference/clibrary - C library - C++ Reference

[8] https://cpp0x.pl/dokumentacja/standard-C/1 - Standard C

[9] https://code.visualstudio.com/ - Visual Studio Code

# 5. Pytania kontrolne

1. Podaj różnice pomiędzy plikami tekstowymi i binarnymi.
2. Scharakteryzuj argumenty funkcji **fwrite()** i **fread()**.
3. Opisz sposób wykrywania końca pliku binarnego.

# 6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.

- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.