Fundusze Europejskie
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Inżynier 5.0 - kształcenie na potrzeby gospodarki
Projekt współfinansowany przez Unię Europejską w ramach programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027.
Nr umowy o dofinansowanie: FERS.01.05-IP.08-0285/23-00.

Wprowadzenie do programowania

Instrukcja do ćwiczenia laboratoryjnego nr 3

Temat: Tablice i operacje na plikach

Politechnika Gdańska
Wydział Elektroniki, Telekomunikacji i Informatyki
Katedra Inżynierii Biomedycznej

Opracował: mgr inż. Antoni Górecki

Spis treści

1 Wprowadzenie

Celem tego laboratorium jest utrwalenie wiedzy z wykładów oraz praktyczne zapoznanie się z tablicami oraz operacjami na plikach.
W ramach przygotowania do realizacji ćwiczenia przeanalizuj poniższe programy oraz wykonaj zadania załączone do nich.

1.1 Tablice

Tablice w języku C są strukturą danych pozwalającą na przechowywanie wielu elementów tego samego typu pod wspólną nazwą. Dostęp do poszczególnych elementów odbywa się za pomocą indeksów, przy czym w C indeksowanie zaczyna się od `0`. Tablice mogą być jedno- i wielowymiarowe, np. jednowymiarowa lista liczb całkowitych albo dwuwymiarowa macierz.

1.1.1 Tablice jednowymiarowe

Tablica jednowymiarowa (1D) to po prostu sekwencja elementów tego samego typu w pamięci. Można ją traktować jako „listę" lub „wektor". Typowym zastosowaniem tablic 1D jest przechowywanie sygnałów, listy wyników, czy danych wczytanych z pliku. Elementy zapisujemy i odczytujemy, podając ich indeks, np. `tab[0]`, `tab[1]`, itd.

W praktyce tablice 1D wykorzystujemy m.in. do:

- przechowywania danych wejściowych,
- obliczania sum, średnich, maksimum, minimum,
- sortowania wartości,
- przekształcania danych (np. progowanie, filtrowanie).

```c
#include <stdio.h>
int main(void) {
int n = 3;
int marks[n]; // deklaracja tablicy 3-elementowej
marks[0] = 10; // wpisanie wartości do elementu
marks[1] = 20;
marks[2] = 30;
printf("Element 0: %d\n", marks[0]);
printf("Element 1: %d\n", marks[1]);
printf("Element 2: %d\n", marks[2]);

printf("\n");

// pętla wyświetlająca całą zawartość tablicy.
// Do iterowania po tablicy najczęściej używamy pętli for.
for(int i=0; i < n; i++)
{
    printf("Element %d z tablicy: %d\n", i+1, marks[i]);
}
```

```
return 0;
}
```

Zadanie: Utwórz tablicę o rozmiarze 5 i przypisz wartości będące kwadratami liczb 0..4. Następnie wypisz zawartość tablicy.

Zadanie: Napisz program, który obliczy maksymalną wartość w tablicy 10-elementowej. Wypełnij ją ręcznie lub losowymi liczbami.

Zamiana elementów miejscami

```c
#include <stdio.h>

int main(void) {
    int tab[4] = {10, 20, 30, 40};
    int temp;


    printf("Oryginalna tablica: \n");
    for (int i = 0; i < 4; i++) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }
    printf("\n");
    // zamiana elementu 1 i 3
    temp = tab[1];
    tab[1] = tab[3];
    tab[3] = temp;


    printf("Tablica po zmianie: \n");
    for (int i = 0; i < 4; i++) {
        printf("tab[%d] = %d\n", i, tab[i]);
    }
    return 0;
}
```

Zadanie: Zamień miejscami pierwszy i ostatni element tablicy o dowolnym rozmiarze.

Sortowanie Najprostsze sortowanie: bąbelkowe.

```c
#include <stdio.h>

#define N 5

int main(void) {
    int tab[N] = {5, 1, 4, 2, 3};
    int temp;
```

```
    printf("Tablica przed sortowaniem:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", tab[i]);
    }

    for (int i = 0; i < N-1; i++) {
        for (int j = 0; j < N-i-1; j++) {
            if (tab[j] > tab[j+1]) {
                temp = tab[j];
                tab[j] = tab[j+1];
                tab[j+1] = temp;
            }
        }
    }
    printf("\n");

    printf("Tablica po sortowaniu:\n");
    for (int i = 0; i < N; i++) {
        printf("%d ", tab[i]);
    }
    return 0;
}
```

Zadanie: Zaimplementuj algorytm sortowania rosnącego dla tablicy 8-elementowej. Spróbuj zmodyfikować kod tak, aby sortował malejąco.

1.1.2 Tablice dwuwymiarowe

Tablica dwuwymiarowa (2D) to zbiór elementów ułożonych w wiersze i kolumny. Można ją traktować jako macierz matematyczną lub prostą reprezentację obrazu cyfrowego (gdzie każda liczba odpowiada jasności piksela).
Deklaracja tablicy 2D ma postać `typ nazwa[wiersze][kolumny]`.
Tablice 2D są szeroko stosowane w informatyce i inżynierii:

- przechowywanie macierzy i wykonywanie operacji algebraicznych (transpozycja, wyznaczniki),
- analiza danych obrazowych (obrazy w skali szarości, maski binarne),
- rozwiązywanie problemów geometrycznych i numerycznych,
- tablice dynamicznych stanów w algorytmach (np. programowanie dynamiczne).

Typowe operacje na macierzach to:

- wypisywanie elementów nad/pod przekątną,
- transponowanie,
- sprawdzanie, czy macierz jest górno- lub dolnotrójkątna,
- progowanie wartości w celu segmentacji (np. obrazu),

- konwersje między reprezentacjami 1D a 2D.

```c
#include <stdio.h>

#define M 2
#define N 3

int main(void) {
    int arr[M][N] = { {5, 3, 5}, {8, 4, 8} };
    int row = 1, col = 2;
    printf("Element wiersz=%d, kolumna=%d to %d\n", row, col, arr[row][col]);

    printf("Cała tablica/macierz: \n");
    for(int i=0; i<M; i++)
    {
        for(int j=0; j<N; j++)
        {
            printf("%d\t", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Zadanie: Utwórz tablicę 3x3 z własnymi wartościami i wypisz element z pozycji [2][0].

Wypisywanie wartości nad i pod przekątną

```c
#include <stdio.h>
#define N 4

int main(void) {
    int arr[N][N] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    for(int i=0; i<N; i++)
    {
        for(int j=0; j<N; j++)
        {
            printf("%d\t", arr[i][j]);
        }
        printf("\n");
    }
```

```
    printf("Elementy nad przekątną:\n");
    for (int i=0; i<N; i++) {
        for (int j=i+1; j<N; j++) {
            printf("%d ", arr[i][j]);
        }
    }
    printf("\n");

    printf("Elementy pod przekątną:\n");
    for (int i=1; i<N; i++) {
        for (int j=0; j<i; j++) {
            printf("%d ", arr[i][j]);
        }
    }
    return 0;
}
```

Zadanie: Dla macierzy 5x5 wypisz elementy na drugiej przekątnej (od prawego górnego do lewego dolnego rogu).

Transponowanie macierzy

```
#include <stdio.h>
#define N 3

int main(void) {
    int A[N][N] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    printf("Macierz oryginalna:\n");
    for (int i=0; i<N; i++) {
        for (int j=0; j<N; j++) {
            printf("%d ", A[i][j]);
        }
        printf("\n");
    }

    int B[N][N];  // macierz transponowana

    for (int i=0; i<N; i++) {
        for (int j=0; j<N; j++) {
            B[j][i] = A[i][j];
        }
    }
```

```
    printf("Macierz transponowana:\n");
    for (int i=0; i<N; i++) {
        for (int j=0; j<N; j++) {
            printf("%d ", B[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Zadanie: Zaimplementuj transponowanie dla macierzy prostokątnej (np. 2x4 → 4x2).

Progowanie wartości Progowanie to zamiana wszystkich wartości poniżej ustalonego progu na 0, a powyżej na 1.

```
#include <stdio.h>
#define N 4
#define M 4

int main(void) {
    int image[N][M] = {
        {0, 2, 5, 8},
        {7, 1, 0, 3},
        {4, 6, 9, 2},
        {1, 8, 2, 5}
    };

    printf("Przed progowaniem \n");
    for (int i=0; i<N; i++) {
        for (int j=0; j<M; j++) {
            printf("%d\t", image[i][j]);
        }
        printf("\n");
    }

    int threshold = 4;

    printf("Po progowaniu (threshold = %d):\n", threshold);
    for (int i=0; i<N; i++) {
        for (int j=0; j<M; j++) {
            if (image[i][j] >= threshold) {
                printf("1 ");
            } else {
                printf("0 ");
            }
        }
        printf("\n");
```

```
33      }
34      return 0;
35  }
```

Zadanie: Napisz program, który policzy, ile pikseli w macierzy obrazu jest większych od zadanej wartości progowej.

1.2 Operacje na plikach

W języku C pliki pozwalają przechowywać dane poza pamięcią programu. Poniżej pokazujemy prosty sposób otwierania, odczytu i zapisu danych w plikach.

1.2.1 Otwieranie i zamykanie plików

```c
#include <stdio.h>

int main(void) {
    FILE *plik;  // wskaźnik do pliku
    plik = fopen("plik.txt", "r"); // otwarcie pliku do odczytu

    if (!plik) {
        printf("Nie mozna otworzyc pliku!\n");
        return 1;
    }

    printf("Plik otwarty pomyslnie!\n");

    fclose(plik);  // zamknięcie pliku
    printf("Plik zamkniety.\n");
    return 0;
}
```

Zadanie: Utwórz plik `dane.txt` z kilkoma liczbami i otwórz go, aby sprawdzić, czy plik został poprawnie otwarty.

1.2.2 Wczytywanie danych z pliku

```c
#include <stdio.h>

int main(void) {
    FILE *plik = fopen("plik.txt", "r");
    if (!plik) {
        printf("Blad przy otwieraniu pliku!\n");
        return 1;
    }

    int liczba;
```

```
    printf("Zawartosc pliku:\n");
    while (fscanf(plik, "%d", &liczba) == 1) {
        printf("%d ", liczba);
    }
    printf("\n");

    fclose(plik);
    return 0;
}
```

Zadanie: Wczytaj liczby z pliku i oblicz ich sumę oraz średnią.

1.2.3 Zapis danych do pliku

```
#include <stdio.h>

int main(void) {
    FILE *plik = fopen("wyniki.txt", "w");
    if (!plik) {
        printf("Blad przy otwieraniu pliku!\n");
        return 1;
    }

    for (int i = 1; i <= 5; i++) {
        fprintf(plik, "Liczba %d kwadrat: %d\n", i, i*i);
    }

    fclose(plik);
    printf("Dane zapisane do pliku wyniki.txt\n");
    return 0;
}
```

Zadanie: Zapisz do pliku wszystkie liczby parzyste od 1 do 20 wraz z ich połową.

1.2.4 Kopiowanie zawartości pliku

```
#include <stdio.h>

int main(void) {
    FILE *we = fopen("dane.txt", "r");
    FILE *wy = fopen("kopia.txt", "w");

    if (!we || !wy) {
        printf("Blad przy otwieraniu plikow!\n");
        return 1;
    }
```

```
    int znak;
    while ((znak = fgetc(we)) != EOF) {
        fputc(znak, wy);
    }

    fclose(we);
    fclose(wy);
    printf("Plik zostal skopiowany.\n");
    return 0;
}
```

Zadanie: Napisz program, który wczytuje plik tekstowy i tworzy nowy plik, w którym każda litera jest zamieniona na wielką.

1.2.5 Przetwarzanie danych w pliku

```
#include <stdio.h>

int main(void) {
    FILE *plik = fopen("dane.txt", "r");
    if (!plik) {
        printf("Nie mozna otworzyc pliku!\n");
        return 1;
    }

    int liczba, suma = 0, count = 0;
    while (fscanf(plik, "%d", &liczba) == 1) {
        if (liczba > 10) {  // liczby większe od 10
            suma += liczba;
            count++;
        }
    }

    fclose(plik);
    printf("Liczb > 10: %d, suma = %d\n", count, suma);
    return 0;
}
```

Zadanie: Napisz program, który policzy, ile liczb w pliku jest większych od 10 i zapisze je do nowego pliku `wieksze10.txt`.

2 Wejściówka

Przykładowe testowe pytania na wejściówkę:

1. Jakie polecenie służy do otwarcia/zamknięcia pliku w języku C?
2. Dana jest tablica dwuwymiarowa. Wskaż element z pozycji [4][7].

3. Określ odpowiednio przeznaczenie każdego trybu otwarcia pliku.
4. Jaki jest indeks ostatniego elementu w tablicy 47-elementowej?
5. Co zostanie wypisane na standardowym wyjściu po wykonaniu poniższych instrukcji?

3 Przed przystąpieniem do zadań

Przed przystąpieniem do zadań należy:

1. Wejść w link z repozytorium GitHub znajdujący się na stronie kursu eNauczanie.
2. Zrobić fork repozytorium zgodnie z instrukcją do Laboratorium 1 (instrukcja na kursie).
3. Sklonować (pobrać) kod z własnego zforkowanego repozytorium za pomocą polecenia:

```
git clone <skopiowany adres z GitHub>
```

W razie problemów proszę wrócić do instrukcji Laboratorium 1.

4 Praca na zajęciach

Ćwiczenia realizowane w oparciu o dwa pliki:
`studenci.txt` – zawiera nazwiska i numery indeksów.
`egzamin.txt` – zawiera numery indeksów oraz punkty uzyskane z egzaminu (max 100 pkt).

Podczas zajęć należy napisać prosty program w języku C, który przetwarza dane studentów. Zakres programu:

- Wczytanie nazwiska i numeru indeksu z pliku `studenci.txt`.
- Wyświetlenie tylko tych studentów, których numer indeksu jest nieparzysty.

Szkielet kodu źródłowego

```c
#include <stdio.h>
#define MAX 50

int main(void) {
    FILE *fp;
    char nazwisko[50];
    int indeks;

    // ----------------------------
    // Krok 1: Wczytanie i wyświetlenie studentów z nieparzystym indeksem
    // ----------------------------
```

```
    fp = fopen(/* ??? */);
    if (!fp) {
        printf("Nie mozna otworzyc pliku studenci.txt\n");
        return 1;
    }

    printf("Studenci z nieparzystym indeksem:\n");
    while (fscanf(fp, "%s %d", nazwisko, &indeks) == 2) {
        if (/* ??? */) {
            printf("%s %d\n", nazwisko, indeks);
        }
    }

    fclose(fp);

    return 0;
}
```

Kolejnym zadania do zrealizowania:

1. Wczytaj wszystkie indeksy studentów do tablicy `indeksy[MAX]`.
2. Posortuj tablicę indeksów rosnąco (np. sortowanie bąbelkowe).
3. Oblicz średnią wartości indeksów i wypisz indeksy większe od średniej.
4. Wczytaj dane z pliku `egzamin.txt`.
5. Na podstawie punktów oblicz oceny (np. >90% = 5, >75% = 4, >50% = 3, reszta 2).
6. Zapisz wyniki do pliku `wyniki.txt` w formacie: ocena i liczba studentów, którzy ją uzyskali.

5 Realizacja zadania wskazanego przez prowadzącego.

Podczas zajęć prowadzący zleca realizację konkretnego zadania związanego z tematem laboratorium w języku C.

6 Literatura pomocnicza i załączniki

- Materiały z wykładów (serwis eNauczanie)