



**POLITECHNIKA LUBELSKA  
WYDZIAŁ ELEKTROTECHNIKI  
I INFORMATYKI**

**KIERUNEK STUDIÓW  
INFORMATYKA**

***MATERIAŁY DO ZAJĘĆ  
LABORATORYJNYCH***

**PROGRAMOWANIE STRUKTURALNE**

**Autor:**  
dr inż. Elżbieta Miłoś

Lublin 2019



**Fundusze  
Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz Społeczny



### Plan zajęć laboratoryjnych:

Lab1.	Wprowadzenie do programowania strukturalnego w języku C. Wprowadzenie do środowiska programistycznego.
Lab2.	Podstawy tworzenia algorytmów, schematy Nassi-Schneidermana.
Lab3.	Proste programy imperatywne. Wprowadzanie i wyprowadzanie danych. Zmienne różnych typów. Instrukcja przypisania. Wyrażenia.
Lab4.	Proste programy strukturalne. Funkcje standardowe i funkcje własne. Argumenty funkcji.
Lab5.	Instrukcje warunkowe IF, IF...ELSE. Operator warunkowy. Instrukcja wyboru SWITCH. Instrukcja BREAK.
Lab6.	Instrukcje iteracyjne WHILE, DO...WHILE. Instrukcja CONTINUE.
Lab7.	Instrukcja iteracyjna FOR.
Lab8.	<i>Kolokwium 1.</i>
Lab9.	Funkcje z argumentami wskaźnikowymi.
Lab10.	Złożone typy danych. Tablice statyczne jedno i wielowymiarowe.
Lab11.	<a href="#">Dynamiczna alokacja pamięci. Tablice dynamiczne.</a>
Lab12.	<a href="#">Łańcuchy znakowe i funkcje łańcuchowe.</a>
Lab13.	<a href="#">Złożone typy danych. Struktury i unie.</a>
Lab14.	<a href="#">Złożone typy danych. Pliki.</a>
Lab15.	<a href="#">Kolokwium 2.</a>

Zajęcia laboratoryjne zawierają pytania kontrolne, które student powinien opanować przed przystąpieniem do zajęć, zestawy 1 lub 2 zadań do analizy (zawierające rozwiązanie w postaci algorytmu NS lub kodu źródłowego w języku C) oraz zestawy 10 zadań (5 podstawowych + 5 dodatkowych) do wykonania (zawierające treść zadania i polecenia do wykonania).

Przygotowanie do zajęć laboratoryjnych może być realizowane przez studenta na podstawie wykładów i zalecanej literatury. Dodatkowo, w każdym laboratorium umieszczono krótkie kompendium wiedzy związane z zakresem realizowanego laboratorium.

Prowadzący zajęcia ocenia przygotowanie merytoryczne studentów do zajęć laboratoryjnych, wybiera zadania do realizacji na laboratorium i ocenia frekwencję i kreatywność studentów na zajęciach.

Zestawy zadań zostały opracowane na podstawie podręcznika: *Montusiewicz J., Miłosz E., Jarońska-Caban M., Podstawy programowania w języku C. Ćwiczenia laboratoryjne*, Wydawca PL, Lublin 2015.

## LABORATORIUM 11. DYNAMICZNA ALOKACJA PAMIĘCI. TABLICE DYNAMICZNE.

### Cel laboratorium:

Zapoznanie z funkcjami przydziału i zwolnienia pamięci. Zastosowanie tych funkcji do typu tablicowego. Wykorzystanie modułów.

### Zakres tematyczny zajęć:

- *tablice statyczne przechowywane w pamięci na stosie,*
- *tablice dynamiczne przechowywane w pamięci na sterckie,*
- *dynamiczna alokacja pamięci,*
- *zwalnianie przydzielonej pamięci,*
- *podział programu na moduły.*

### Kompendium wiedzy:

#### **Tablice (C89):**

- **statyczna** - rozmiar tablicy jest znany na etapie kompilacji programu, zmienna przechowywana w pamięci na tzw. stosie (powstaje, gdy program wchodzi do bloku, w którym zmienna jest zadeklarowana, a zwalniana pamięć jest w momencie, kiedy program opuszcza ten blok), liczba elementów tablicy nie ulega zmianie w trakcie działania programu,
- **dynamiczna** - rozmiar tablicy jest ustalany na etapie wykonania programu, zmienna przechowywana w pamięci na tzw. sterckie (obszar pamięci wspólny dla całego programu, gdzie przechowywane są zmienne, których czas życia nie jest związany z poszczególnymi blokami).

**Tablice VLA (variable length array) (C99):** rozmiar tablicy może być zdefiniowany w trakcie wykonania programu - może być określony wartością zmiennej, wartość ta nie musi być znana na etapie kompilacji, raz stworzona tablica zachowuje swój rozmiar.

#### **Funkcje dynamicznej alokacji pamięci z biblioteki *stdlib.h*:**

```
void *malloc(size_t n);  
void *calloc(size_t n, size_t size);  
void *realloc(void *ptr, size_t size);
```

Funkcje przydzielają na sterckie wolną pamięć.

Zwracają wskaźnik do **void**. Jeśli nie jest możliwe przydzielenie pamięci, to funkcje zwracają wskaźnik **NULL**. **size\_t** jest odwołaniem do pewnego typu danych (typ całkowity bez znaku).

Parametrem funkcji **malloc** jest liczba przydzielonych bajtów pamięci.

Parametrami funkcji **calloc** są: liczba komórek pamięci i rozmiar każdej komórki w bajtach. Pamięć przydzielana funkcją **calloc** jest inicjowana zerami.

Funkcja **realloc** zmienia rozmiar przydzielonego wcześniej bloku pamięci wskazywanego przez **ptr** do rozmiaru **size** bajtów.



Bezpośrednie nadanie wartości wskaźnika do **void** wskaźnikowi innego typu nie jest błędem, ale ze względu na czytelność zalecane jest użycie jawnego rzutowania.

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

```
double*tab; int n=10;
tab=(double*)malloc(n*sizeof(double));
if(tab==NULL)
    {printf("Brak wolnej pamieci"); exit(1);}
```

Funkcja **free** zwalnia obszar pamięci alokowany dla wskaźnika będącego jej parametrem:

```
void free(void *ptr);
free (tab);
```

**Przykład 1. Dynamiczna alokacja pamięci dla tablicy jednowymiarowej:**

```
int *t, i;
t = (int *) malloc(10 *sizeof(int));
if (t==NULL)
{printf (" brak wolnej pamięci "); exit(1);}
for (i=0; i<10; i++)
    {*(t + i) = i * i;
      printf("\n%d", *(t + i));
    }
free (t);
```

Funkcja **malloc** przydziela pamięć dla tablicy 10 liczb całkowitych. Jeśli funkcja zwróci **null**, czyli 0, nastąpi wyjście z programu. Jeśli przydzielenie pamięci powiedzie się, to w pętli **for** będą obliczone i wyświetlone elementy tablicy **t**: kwadraty liczb od 0 do 9. Następnie pamięć zajęta przez tablicę **t** będzie zwolniona.

**Przykład 2. Dynamiczna alokacja pamięci dla tablicy dwuwymiarowej:**

```
int **p, i;
p =(int*) malloc(sizeof(int *)*5);
for(i=0; i < 5; i++)
p[i] =(int*) malloc(sizeof(int)*10);
```

Deklaracja dynamiczna tablicy **p** liczb całkowitych składającej się z 5 wierszy i 10 kolumn. Wiersz jest tutaj traktowany jak tablica jednowymiarowa zawierająca 10 liczb, dlatego przydział pamięci wykonany jest w pętli dla każdego wiersza.

**Podział programu na moduły:**

**Moduł**- fragment programu kompilowany jako jeden plik składający się z:

- **interfejsu** (co plik udostępnia) – pliki nagłówkowe (\*.h)
- **implementacji** (definicje realizacji) – pliki źródłowe (\*.c)

**module.h**

```
//tu deklaracje funkcji
```

**module.c**

```
#include "module.h"
```

```
//tu definicje funkcji
```



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



### Pytania kontrolne:

1. Kiedy należy stosować tablice dynamiczne?
2. Jak dynamicznie alokować pamięć dla tablicy?
3. Jak zwolnić pamięć alokowaną dynamicznie?
4. Jak obsłużyć błąd alokacji pamięci?
5. Jak podzielić program na moduły?
6. Jakie są zalety podziału programu na moduły?

### Zadania do analizy

#### Zadanie 11.1. Tablice dynamiczne jednowymiarowe

- Przeanalizuj przykład programu wykorzystującego tablicę dynamiczną jednowymiarową.  
Funkcja `utworzTD` tworzy jednowymiarową tablicę dynamiczną o `n` elementach typu `int` i wczytuje dane do tablicy z klawiatury.  
Funkcja `wyswietlTD` wyświetla tablicę `n` elementową liczb całkowitych.  
Funkcja `usunTD` zwalnia pamięć przydzieloną tablicy.
- Podaj tekst w komentarzach.

#### Main.c

```
1  #include <stdio.h>      //???
2  #include <stdlib.h>
3
4  int *utworzTD(int n);    //???
5  void wyswietlTD(int * tabD, int n); //???
6  void usunTD(int *tabD); //???
7  //=====
8  int main(int argc, char *argv[]) { //???
9      int ile; int *tab1;
10     //tablice dynamiczne
11     printf("Podaj liczbę elementów tablicy ");
12     scanf("%d",&ile);
13     tab1=utworzTD(ile);    //???
14     printf("tablica dyn:\n");
15     wyswietlTD(tab1, ile); //???
16     usunTD(tab1);         //???
17
18     return 0;
19 }
20 //=====
21 int *utworzTD(int n)      //???
22 {
23     int i;
24     int *tabD = (int*) malloc (n *sizeof(int)); //???
25     if (tabD==NULL)       //???
26         {printf("Bład przydziału pamieci\n");
```



```
26     exit(EXIT_FAILURE); }
27
28     for (i=0; i<n; i++)
29     {     printf("wpisz liczbe tab[%d]: ", i);
30           scanf("%d", tabD + i);
31     }
32     return tabD;           //???
33 }
34 //-----
35 void wyswietlTD(int * tabD, int n)           //???
36 { int i;
37   printf ("\nZawartosc tablicy:\n");
38   for (i=0; i<n; i++)
39     printf ("%d\t", *(tabD+i));           //???
40   printf ("\n ");
41 }
42 //-----
43 void usunTD(int *tabD)           //???
44 { free(tabD);                   //???
45   tabD=0;
46 }
```

### **Zadanie 11.2. Tablice dynamiczne jednowymiarowe – podział na moduły**

- Przeanalizuj przykład programu wykorzystującego tablicę dynamiczną jednowymiarową z podziałem programu na moduły. Porównaj zawartość i kod projektu z poprzednim zadaniem.
- Podaj tekst w komentarzach.

```
Main.c           //???
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "TabliceDyn.h"           //???
4
5  int main(int argc, char *argv[]) {
6     int ile; int *tab1;
7     //tablice dynamiczne
8     printf("Podaj liczbe elementów tablicy ");
9     scanf("%d",&ile);
10    tab1=utworzTD(ile);           //???
11    printf("tablica dyn:\n");
12    wyswietlTD(tab1, ile);           //???
13    usunTD(tab1);                 //???
14
15    return 0;
16 }
```



```
TabliceDyn.h    //???
1  int *utworzTD(int n);    //???
2  void wyswietlTD(int * tabD, int n);    //???
3  void usunTD(int *tabD);    //???

TabliceDyn.c    //???
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include<time.h>
4  #include "TabliceDyn.h"    //???
5  //tablice dynamiczne jednowymiarowe
6  int *utworzTD(int n)    //???
7  { int i;
8    int *tabD = (int*) malloc (n *sizeof(int));
9    if(tabD==NULL)
10       {printf("Blad przydzialu pamieci\n");
11         exit(EXIT_FAILURE);
12       }
13    for (i=0; i<n; i++)
14    {    printf("wpisz liczbe tab[%d]: ", i);
15         scanf("%d", tabD + i);
16    }
17    return tabD;
18 }
19 //-----
20 void wyswietlTD(int * tabD, int n)    //???
21 { int i;
22    printf ("\nZawartosc tablicy:\n");
23    for (i=0; i<n; i++)
24    printf ("%d\t", *(tabD+i));
25    printf ("\n ");
26 }
27 //-----
28 void usunTD(int *tabD)    //???
29 { free(tabD);
30   tabD=0;
31 }
```

### Zadanie 11.3. Tablice dynamiczne dwuwymiarowe

- Przeanalizuj kody funkcji wykorzystujących tablicę dynamiczną dwuwymiarową. Funkcja `utworzT2D` tworzy dwuwymiarową tablicę dynamiczną o  $n$  wierszach i  $m$  kolumnach elementów typu `int` i wczytuje dane do tablicy z klawiatury. Funkcja `wyswietlT2D` wyświetla tablicę liczb całkowitych o rozmiarze  $n \times m$ .
- Podaj tekst w komentarzach.



```
//w jakim pliku umieścić ten kod?
1  int** utworzT2D(int n, int m)          //???
2  {
3      int**tab2D; int i,j;
4      tab2D = (int*) malloc(sizeof(int *)*n); //???
5      if(tab2D==NULL)                    //???
6          {printf("Blad przydzialu pamieci\n");
7            exit(EXIT_FAILURE);
8          }
9      for(i=0; i < n; i++)
10     {
11         tab2D[i] = (int*) malloc(sizeof(int)*m); //???
12         if(tab2D[i]==NULL)                    //???
13             {printf("Blad przydzialu pamieci\n");
14               exit(EXIT_FAILURE);
15             }
16     }
17     for(i=0;i<n;i++)
18     {
19         for(j=0;j<m;j++)
20             {// *(tab2D+i)+j)=i*j;          //???
21               printf("tab[%d][%d]= ", i,j);
22               scanf("%d", *(tab2D+i)+j); //???
23             }
24     }
25     return tab2D;                          //???
26 }
27 //-----
28 void wyswietlT2D(int**tab2D,int n, int m) //???
29 {
30     int i,j;
31     printf("Tablica dynamiczna 2D\n");
32     for(i=0;i<n;i++)
33     {
34         for(j=0;j<m;j++)
35         {
36             printf("%d\t",*(tab2D+i)+j)); //???
37         }
38         printf("\n");
39     }
40 }
```

```
//w jakim pliku umieścić ten kod?
1  int**tab2D; int i,j,n,m;              //???
2  printf("\ntablica 2D\n");
3  printf("Podaj liczbę wierszy tablicy ");
4  scanf("%d",&n);
5  printf("Podaj liczbę kolumn tablicy ");
6  scanf("%d",&m);
```





7	tab2D = utworzT2D(n,m);	//???
8	wyswietlT2D(tab2D,n,m);	//???
9	usunTD(tab2D);	//???

### Zadania do wykonania

#### **Zadanie 11.4. Program z modułem zawierający funkcje na tablicach dynamicznych**

Napisz program operujący na tablicach dynamicznych jedno i dwuwymiarowych. Projekt powinien zawierać pliki: `main.c`, `TabliceDyn.h`, `TabliceDyn.c`.

W plikach `TabliceDyn.h` i `TabliceDyn.c` umieść deklaracje i definicje funkcji pozwalających na:

- tworzenie, wyświetlenie i usunięcie jednowymiarowej tablicy dynamicznej *liczb rzeczywistych*,
- tworzenie i wyświetlenie dwuwymiarowej tablicy dynamicznej *liczb rzeczywistych*.

W pliku `main.c` przetestuj napisane funkcje.

#### **Zadanie 11.5. Tablica statyczna i dynamiczna**

Napisz program, który:

- zapełni 100 elementową tablicę statyczną `tabS` wylosowanymi liczbami od 1 do 100 (funkcja `losuj`),
- obliczy, ile z nich jest z podanego przedziału  $\langle a, b \rangle$ , a następnie utworzy tablicę dynamiczną odpowiedniego rozmiaru i zapełni ją tymi liczbami (funkcja `nowatabDyn` z parametrami: wskaźnik do tablicy statycznej `tabS`, liczba jej elementów `n`, przedziały `a` i `b`, wskaźnik do liczby elementów tablicy dynamicznej `m`; funkcja zwraca wskaźnik do tablicy dynamicznej).

Wyświetl obydwie tablice. Wykorzystaj odpowiednie funkcje.

#### **Zadanie 11.6. Tworzenie tablic dynamicznych z tablicy dwuwymiarowej**

Dane: `tab2` - tablica liczb rzeczywistych o wymiarach `n` wierszy, `m` kolumn.

Napisz funkcje:

- `f1` - tworzy tablicę `tabD`, zawierającą elementy dodatnie,
- `f2` - tworzy tablicę `tabU`, zawierającą elementy ujemne.

Napisz program, w którym wczytane są dane, wywołane funkcje, wyświetlone wyniki.

### Zadania dodatkowe

#### **Zadanie 11.7. Maraton**

Napisz program ewidencjonujący czasy osiągnięte przez zawodników maratonu i wyszukujący zwycięzcę. Liczba zawodników `n` nie jest dokładnie znana, zakłada się, że startowa pula numerów jest ograniczona do 300 osób. Należy ewidencjonować dokładnie tyle



czasów ile zawodników. Program powinien wyszukać najlepszy wynik i wyświetlić numer startowy/ numery osoby/ osób z tym wynikiem. Wykorzystaj odpowiednie funkcje.

### **Zadanie 11.8. Obliczenie sum w wierszach i kolumnach tablicy dwuwymiarowej**

Napisz funkcję obliczającą sumy w wierszach i sumy w kolumnach w dwuwymiarowej tablicy liczb rzeczywistych i zwracającą wyniki, jako dwie tablice. Rozmiary ( $n$ ,  $m$ ) i elementy danej tablicy `tab` są parametrami przekazanymi do funkcji. Tablica sum w kolumnach `sumaK` jest przekazana z funkcji jako parametr wskaźnikowy, wskaźnik do utworzonej tablicy sum w wierszach `sumaW` jest przekazany poprzez return.

Prototyp funkcji może wyglądać następująco:

```
float * sumaW(int n, int m, float tab[][m], float * sumaK);
```

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

### **Zadanie 11.9. Tworzenie tablicy dynamicznej dwuwymiarowej**

W hurtowni jest  $n$  towarów. Tablica `dane` zawiera w kolumnie zerowej ceny, w kolumnie pierwszej ilości towarów. Napisz funkcję tworzącą tablicę informacji o towarach (cena, ilość), których wartość jest większa od liczby podanej przez użytkownika. Tablice są deklarowane dynamicznie.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

### **Zadanie 11.10. Tablica dynamiczna zwracana przez wskaźnik**

Tablica `punkty` zawiera współrzędne  $n$  punktów na płaszczyźnie. Napisz funkcję tworzącą tablicę odległości punktów od początku układu współrzędnych. Wskaźnik do tablicy jest przekazany z funkcji przez return.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

### **Zadanie 11.11. Moduł z funkcjami do obsługi tablic**

Utwórz funkcje wykonujące podstawowe operacje na tablicach jedno i dwuwymiarowych o podanych rozmiarach i zapisz je w module `tablice.h`. Parametrami funkcji powinny być rozmiary i tablica. Podstawowe operacje (oprócz wczytania i wyświetlenia zawartości) to:

- sumowanie wszystkich elementów tablicy,
  - obliczanie średniej arytmetycznej elementów tablicy,
  - szukanie wartości minimalnej i maksymalnej,
  - zliczanie elementów o podanej wartości,
  - wykonanie operacji na wybranych elementach tablicy.
- Przetestuj te funkcje na utworzonych tablicach dynamicznych.

## LABORATORIUM 12. ŁAŃCUCHY ZNAKOWE I FUNKCJE ŁAŃCUCHOWE.

### Cel laboratorium:

Zapoznanie z metodami inicjalizacji łańcuchów znakowych i funkcjami działającymi na łańcuchach znakowych.

### Zakres tematyczny zajęć:

- *pojęcie łańcucha znaków,*
- *stałe i zmienne łańcuchowe,*
- *wczytywanie i wyprowadzanie łańcuchów,*
- *funkcje standardowe na łańcuchach z biblioteki string.h.*

### Kompendium wiedzy:

**Łańcuch znakowy:** tablica typu char zakończona znakiem zerowym (\0).

#### Definiowanie /inicjacja łańcuchów:

- **Stała łańcuchowa** (literal łańcuchowy) - sekwencja znaków w "".
- **Zmienna łańcuchowa** - inicjacja tylko w definicji.

```
#define FIRMA "ABC"
#define DL 20

char imie[10];
char nazwisko[DL];
char dyrektor[]="Kowalski"; //char *dyrektor="Kowalski";
char prezes[]={ 'K', 'o', 's', '\0' };
printf("Dyrektorem firmy \" %s\" jest %s a prezesem %s\n",
      FIRMA, dyrektor, prezes);
```

#### Wczytywanie łańcuchów - funkcje: scanf(), gets(), fgets():

- Funkcja **scanf()** służy do wczytywania różnych zmiennych, w tym łańcuchów, pobierając znaki ze standardowego urządzenia wejścia do chwili napotkania znaku niedrukowanego (nowej linii '\n', tabulatora, spacji), który pozostaje. Funkcja ta dodaje na końcu łańcucha znak '\0'.

```
scanf("%s", tablica_znakowa);
```

```
scanf("%s", nazwisko);
```

- Funkcja **gets()** służy do wczytywania łańcuchów pobierając znaki (w tym spacje) ze standardowego urządzenia wejścia do chwili napotkania znaku nowej linii '\n' (ale bez niego), dodaje na końcu znak '\0' i przekazuje łańcuch do programu. Nie sprawdza liczby wprowadzanych znaków z zadeklarowanym rozmiarem tablicy.

```
gets(tablica_znakowa);
```

```
gets(imie);
```



<ul style="list-style-type: none"> <li>Funkcja <b>fgets()</b> umożliwia podanie liczby znaków, które zostaną wprowadzone. Po odczytaniu znaku '\n' pozostawia go w łańcuchu.</li> </ul>
<b>fgets(nazwa_tablicy, liczba_znakow, strumien_wej);</b> fgets(nazwisko, DL, stdin);
<p><b>Wyprowadzanie łańcuchów</b> - funkcje: printf(), puts() fputs():</p> <ul style="list-style-type: none"> <li>Funkcja printf() służy do wyprowadzania wartości różnych zmiennych, w tym łańcuchów, na standardowe urządzenie wyjścia i można ją użyć w następujących postaciach (znak \n należy dodać do łańcucha formatującego, by przejść do nowej linii):</li> </ul>
<b>printf(stala_lancuchowa);</b> <b>printf("%s\n", tablica_znakowa);</b> <b>printf("%ps\n", tablica_znakowa);</b>
<p>gdzie: p - minimalna liczba znaków wyprowadzanego pola, jeżeli p jest ze znakiem minus, zmienna w polu wyrównywana jest do lewej.</p> <p>printf("Witaj w swiecie programowania ");                  printf("%s\n", nazwisko);                  printf("Witaj %20s\n", nazwisko);</p> <ul style="list-style-type: none"> <li>Funkcja puts() wyprowadza łańcuch zakończony znakiem '\0' i do wyprowadzanego łańcucha dodaje znak końca linii:</li> </ul>
<b>puts(stala_lancuchowa);</b> <b>puts(tablica_znakowa);</b>
<p>puts(FIRMA);                  puts(nazwisko);</p> <ul style="list-style-type: none"> <li>Funkcja fputs() wyprowadza łańcuch na wskazane wyjście (plik lub ekran - stdout). Przy wyświetleniu fputs() nie dodaje znaku przejścia do nowego wiersza</li> </ul>
<b>fputs(tablica_znakowa, stdout);</b>
fputs(nazwisko, stdout);
<p><b>Funkcje łańcuchowe z biblioteki string.h:</b></p> <ul style="list-style-type: none"> <li>Operacja podstawienia (kopiowania):                      Poza inicjalizacją łańcuchów w chwili definiowania ich typów, napisów nie można podstawiać do tablic przez zastosowanie znaku '='. Należy użyć kopiowania łańcuchów:</li> </ul>
<b>strcpy(dokad, skad);</b> <b>strncpy(dokad, skad, maxliczba);</b>
<p>gdzie: dokad – tablica znakowa lub napis,                      skad – napis lub stała napisowa,                      maxliczba - maksymalna liczba znaków do skopiowania.</p> <p>strcpy(nazwisko, "Nowak");                  strncpy(nazwisko, "Nowak", DL-1)</p>

<ul style="list-style-type: none"> <li>Operacja łączenia (konkatenacja): Funkcja dołącza do dotychczasowej zawartości napis1 zawartość napis2, pierwszy argument zmienia się, drugi pozostaje bez zmian.</li> </ul>
<b>strcat (napis1, napis2);</b> <b>strncat (napis1, napis2, maxliczba);</b>
<pre>strcat(nazwisko, imie); max= DL-strlen(imie)-1 strncat(nazwisko, imie,);max);</pre>
<ul style="list-style-type: none"> <li>Porównywanie napisów: Funkcja zwraca wartość 0, gdy napisy są równe leksykograficznie (słownikowo), wartość dodatnią, gdy napis1 jest słownikowo większy (leży dalej w słowniku), wartość ujemną, gdy napis1 jest mniejszy (leży bliżej początku słownika). Przy porównywaniu funkcja korzysta z kodów ASCII (cyfry &lt; duże litery &lt; małe litery).</li> </ul>
<b>strcmp (napis1, napis2);</b> <b>strncmp (napis1, napis2, maxliczba);</b>
<pre>strcmp(dyrektor, nazwisko); strncmp(dyrektor, nazwisko,5);</pre>
<ul style="list-style-type: none"> <li>Obliczanie długości łańcucha znaków: Funkcja oblicza długość łańcucha znaków od pierwszego znaku do znaku terminalnego '\0'.</li> </ul>
<b>strlen (napis1);</b>
<ul style="list-style-type: none"> <li>Konwersja tekstu na liczbę:</li> </ul>
<b>atoi(napis); //int</b> <b>atol(napis); //long</b> <b>atof(napis); //double</b>
<pre>int i=atoi("1234"); double x=atof("56.78"); printf("%d\n",i); printf("%f\n",x);</pre>

### Pytania kontrolne:

1. Co to jest łańcuch? Czym różni się zapis 'x' od "x"?
2. Jak zdefiniować stałą, a jak zmienną łańcuchową?
3. Podaj funkcje do wczytywania łańcuchów i ich zastosowania.
4. Podaj funkcje do wyświetlania łańcuchów i ich zastosowania.
5. Jak obliczyć długość łańcucha?
6. Jak skopiować łańcuchy?
7. Jak połączyć dwa łańcuchy w jeden łańcuch?
8. Jak porównać łańcuchy?



### Zadania do analizy

#### Zadanie 12.1. Operacje na łańcuchach

- Przeanalizuj przykład programu wykorzystującego łańcuchy.
- Podaj tekst w komentarzach.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>    //???
4  #include <locale.h>
5  #define FIRMA "ABC"
6
7  int main(int argc, char *argv[]) {
8  setlocale(LC_ALL, "");    //???
9  char dyrektor[]="Kowalski"; //char *dyrektor="Kowalski";
10 char prezes[]={ 'K', 'o', 's', '\0' }; //???
11 printf("Dyrektorem firmy \" %s\" jest %s a prezesem %s\n",
FIRMA, dyrektor, prezes);    //???
12 char str1[5], str2[20], str3[7], str4[20], str5[20],
str6[50];
13 printf("Kto jest autorem Pana Tadeusza\n");
14 gets(str1);    //???
15 puts (str1);    //???
16 printf("Kto jest autorem Akademii Pana Kleksa\n");
17 scanf("%s",str2);    //???
18 printf ("Autor: %s znany i lubiany\n",str2);    //???
19 printf ("Autor: %20s znany i lubiany\n",str2);    //???
20 printf ("Autor: %-20s znany i lubiany\n",str2);    //???
21 printf("=====\n");
22 printf("Kto jest autorem Pana Tadeusza ");
23 fgets(str3,7,stdin);    //???
24 fputs(str3,stdout);    //???
25 printf("=====\n");
26 strcpy(str4, "Witam ");    //???
27 printf ("%s\n", str4);
28 strcpy(str5, "tutaj");    //???
29 printf("%s\n",str5);
30 printf ("Konkatenacja str4 i str5:\n");
31 strcat(str4, str5);    //???
32 printf ("%s\n", str4);
33 printf("Kopiowanie str4 do str6:\n");
34 strcpy(str6, str4);
35 printf ("%s\n", str6);
36 printf("Dlugosc str6:%d\n",strlen(str6));    //???
37 return 0;
38 }
```



## Zadanie 12.2. Kopiowanie łańcuchów

- Przeanalizuj przykład programu wykorzystującego łańcuchy.
- Podaj tekst w komentarzach.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>          //???
4  #define DL 30
5  #define MAXDL 6
6  #define N 3
7
8  int main(int argc, char *argv[]) {
9  char slowa[3][MAXDL];        //???
10 char temp[DL];               //???
11 int i=0;
12 printf("Podaj %d slow zaczynajacych sie na a:\n",N);
13 while(i<N && gets(temp))    //???
14 { if(temp[0] != 'a')         //???
15     printf("%s nie zaczyna sie na a!\n", temp);
16     else
17     {strncpy(slowa[i], temp, MAXDL-1); //???
18       slowa[i][MAXDL-1]='\0';         //???
19       i++;
20     }
21 }
22 printf("Przyjete slowa:\n")   ;
23 for(i=0;i<N;i++)
24     puts(slowa[i]);           //???
25
26 printf("Podaj %d slowa do skrocenia\n",N);
27 while(i>0 && gets(temp))
28 {
29     if(strlen(temp)>MAXDL)      //???
30         *(temp+MAXDL)='\0';    //???
31     puts(temp);
32     i--;
33 }
34 return 0;
35 }
```

### Zadania do wykonania

## Zadanie 12.3. Wyśrodkowanie tekstu

Napisz funkcję pozwalającą na wyśrodkowywanie podanego tekstu. Przyjmij, że liczba znaków w linii wynosi 80, zaś tekst jest wprowadzany bez kontrolowania długości wprowadzanych znaków.





Zastosuj tę funkcję do wyświetlenia:

- nazwy przedmiotu, zdefiniowanego jako stała łańcuchowa,
- twoich danych osobowych (imię i nazwisko), zainicjowanych podczas definicji,
- danych kolegi (imię i nazwisko), wczytanych z klawiatury.

#### **Zadanie 12.4. Formatowanie wiersza**

Napisz funkcje pozwalające na wyrównywanie do lewej i wyrównanie do prawej tekstu.

Zastosuj te funkcje do wyświetlenia twojego imienia i nazwiska.

Zastosuj te funkcje do wyświetlenia wiersza *Lokomotywa* Juliana Tuwima:

*Stoi na stacji lokomotywa,  
Ciężka, ogromna i pot z niej spływa:  
Thusta oliwa.  
Stoi i sapie, dyszy i dmucha,  
Żar z rozgrzanego jej brzucha bucha:  
Buch - jak gorąco!  
Uch - jak gorąco!*

Przyjmij, że liczba znaków w linii wynosi 80, zaś tekst jest wprowadzany bez kontrolowania długości wprowadzanych znaków. Wiersz zapisz w jednej tablicy dwuwymiarowej z zainicjowanymi wartościami.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

#### **Zadanie 12.5. Obliczanie długości napisu**

Napisz program obliczający długość zdania powstałego przez konkatencję następujących stałych łańcuchowych: "Biblioteka", "string.h", "pozwala", "na", "zastosowanie", "funkcji", "do", "łańcuchow", "znakowych". Pamiętaj o wprowadzeniu spacji, aby powstałe zdanie wyglądało właściwie.

#### **Zadanie 12.6. Zamiana tekstu na duże litery**

Napisz funkcję pozwalającą na zamianę wprowadzonego tekstu na tekst, który będzie wypisany dużymi literami. Wprowadzony tekst może zawierać duże, małe litery oraz inne znaki. Do tego celu zastosuj funkcję kontrolującą liczbę wprowadzonych znaków. Wypisz napis wprowadzony i zmieniony. Wykorzystując funkcję z zadania 12.2, wypośrodkuj zmieniony napis.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

#### **Zadania dodatkowe**

#### **Zadanie 12.7. Zakończenie działania programu**

Napisz funkcję, w której zakończenie programu nastąpi po wprowadzeniu napisu "quit". Program pozwala na wprowadzanie wielu napisów (pętla nieskończona).

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.





### **Zadanie 12.8. Słownikowe porządkowanie wyrazów**

*Wersja podstawowa:*

Napisz funkcję, która uporządkuje leksykograficznie (od litery a do z) pięć trzyliterowych wyrazów wprowadzonych z klawiatury, uwzględniając tylko pierwszą literę wyrazów.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

*Wersja zaawansowana:*

Napisz funkcję, która uporządkuje leksykograficznie (od litery a do z) pięć trzyliterowych wyrazów wprowadzonych z klawiatury, uwzględniając kolejne litery wyrazów.

Napisz program, w którym wczytane są dane, wywołana funkcja, wyświetlone wyniki.

### **Zadanie 12.9. Tworzenie napisów w odbiciu lustrzanym**

Napisz funkcję, pozwalającą odwrócić kolejność znaków wprowadzonego napisu. W wersji rozbudowanej zastosuj funkcję zamieniającą napis na duże litery (zadanie 12.5), oraz wyśrodkowujący go (zadanie 12.2, liczba znaków w linii 80).

Napisz program, w którym wczytane są dane, wywołane funkcje, wyświetlone wyniki.

### **Zadanie 12.10. Szyfr Cezara**

Napisz funkcję, która pozwoli na szyfrowanie wprowadzonego tekstu stosując szyfr Cezara. Zastosuj przesunięcie znaków o wartość 3. Zakończenie wprowadzania napisów nastąpi po wpisaniu napisu "quit". Wypisz napisy zaszyfrowane.

Napisz drugą funkcję dekodującą teksty zaszyfrowane kodem Cezara.

### **Zadanie 12.11. Moduł z funkcjami do obsługi łańcuchów**

Zgrupuj wszystkie własne funkcje do obsługi łańcuchów w module `lancuchy.h`. Napisz program, w którym przetestujesz funkcje z tego modułu.



## LABORATORIUM 13. ZŁOŻONE TYPY DANYCH. STRUKTURY I UNIE.

### Cel laboratorium:

Zaznajomienie z wykorzystaniem struktur i unii, jako typów złożonych do organizacji skomplikowanych danych. Nabycie praktycznych umiejętności wyboru dobrego sposobu reprezentacji danych oraz programowania z zastosowaniem struktur złożonych.

### Zakres tematyczny zajęć:

- *pojęcie struktury,*
- *pojęcie unii,*
- *sposoby deklaracji struktur i unii,*
- *inicjalizacja struktur,*
- *dostęp do pól struktur i unii za pomocą zmiennej,*
- *dostęp do pól struktur i unii za pomocą wskaźników,*
- *typowe operacje na strukturach,*
- *struktury jako parametry funkcji.*

### Kompendium wiedzy:

Pojęcie **struktura** można interpretować jako typ lub jako zmienną. Struktura, jako typ danych, składa się z pól i pozwala na przechowywanie danych różnych typów. Struktura, jako zmienna, to zmienna typu strukturalnego.

#### *Deklaracja struktury: typ z nazwą + zmienna:*

```
struct nazwaTypu {    //typ
    typPola1 nazwaPola1;
    typPola2 nazwaPola2;
    ...
};
struct nazwaTypu nazwazmiennej; //zmienna
```

```
struct student{
    char nazwisko[25];
    int wiek;
};
struct student st1;
```

#### *Definicja struktury: deklaracja + inicjalizacja:*

```
struct nazwaTypu zmienna={wartP1, wartP2,..};
struct student st2={"Kowalski",20};
```

#### *Dostęp do składowych (pola struktury) za pomocą zmiennej:*

```
zmienna.nazwaPola
```

```
st1.nazwisko
```



**Dostęp do składowych (pola struktury) za pomocą wskaźnika do struktury:**

```
struct nazwaTypu *wskaźnik;  
wskaźnik->nazwaPola //dostęp do pola
```

```
struct student st3={"Kos",25};  
struct student *wsk=&st3;  
printf("%d \n", st3.wiek);  
printf("%d \n", wsk->wiek);  
printf("%d \n", (*wsk).wiek);
```

**Unia** – typ złożony do przechowywania różnych rodzajów danych w tym samym obszarze pamięci (jednak nie równocześnie).

Definicja unii jest analogiczna jak struktury tylko z innym słowem kluczowym:

```
union nazwaTypu { //typ  
    typPola1 nazwaPola1;  
    typPola2 nazwaPola2;  
    ...  
};  
union nazwaTypu nazwazmiennej; //zmienna
```

**Dostęp do składowych (pola unii):**

```
zmienna.nazwaPola //za pomocą zmiennej  
lub  
union nazwaTypu *wskaźnik; //deklaracja wskaźnika  
wskaźnik->nazwaPola //za pomocą wskaźnika
```

```
union magazyn {  
    int calkowita;  
    double rzeczywista;  
    char litera; };  
union magazyn koszyk;  
koszyk.calkowita=22; //22 zapisane w koszyk, zajęte 2 bajty  
koszyk.rzeczywista=2.0; //22 usunięte, 2.0 zapisane,  
                        //zajęte 8 bajtów  
koszyk.litera='a'; // 2.0 usunięte, 'a' zapisane, zajęty 1 bajt
```

**Pytania kontrolne:**

1. Co to jest struktura?
2. Co to jest unia, czym różni się od struktury?
3. Jak odwołać się do pola struktury za pomocą zmiennej?
4. Jak odwołać się do pola struktury za pomocą wskaźnika?
5. Jak zainicjalizować pola struktury wartościami?
6. Jak wczytać i wyświetlić strukturę lub unię?
7. Czym różni się funkcja, w której parametrem jest struktura, od funkcji, w której parametrem jest wskaźnik na strukturę?



### Zadania do analizy

#### Zadanie 13.1. Dane osobowe – przetwarzanie struktur

- Przeanalizuj przykład programu wykorzystującego struktury. Program wczytuje i wyświetla dane osobowe dwóch osób z wykorzystaniem funkcji z parametrem typu struktura lub wskaźnik na strukturę.
- Podaj tekst w komentarzach.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct daneos {                               //???
4      char imie[15];
5      char nazwisko[25];
6  };
7  struct daneos wczytaj();                       //???
8  void wyswietl(struct daneos os);               //???
9
10 void wczytaj2(struct daneos *wsk);             //???
11 void wyswietl2(struct daneos *wsk);            //???
12 //=====
13 int main(int argc, char *argv[])
14 { struct daneos osoba, osoba2;                 //???
15   printf("Osoba 1\n");
16   osoba=wczytaj();                             //???
17   wyswietl(osoba);                             //???
18   printf("Osoba 2\n");
19   wczytaj2(&osoba2);                           //???
20   wyswietl2(&osoba2);                          //???
21   system("PAUSE");
22   return 0;
23 }
24 //=====
25 struct daneos wczytaj()                        //???
26 { struct daneos os;
27   printf("Podaj imie "); gets(os.imie);         //???
28   printf("Podaj nazwisko "); gets(os.nazwisko);
29   return os;                                   //???
30 }
31 void wyswietl(struct daneos os)                 //???
32 {
33   printf("%s %s \n", os.imie, os.nazwisko); //???
34 }
35 void wczytaj2(struct daneos *wsk)               //???
36 { printf("Podaj imie ");
37   gets(wsk->imie);                             //???
38   printf("Podaj nazwisko ");
39   gets(wsk->nazwisko);
40 }
```



```

41 void wyswietl2(struct daneos *wsk)           //???
42 {
43     printf("%s %s \n",  wsk->imie,wsk->nazwisko); //???
44 }

```

### Zadanie 13.2. Dane klienta – przetwarzanie struktur i unii

- Przeanalizuj przykład programu wykorzystującego struktury i unie.  
Program wczytuje i wyświetla dane osobowe dwóch klientów: jeden to osoba fizyczna, opisana imieniem, nazwiskiem i peselem, drugi to firma opisywana nazwą. Program wykorzystuje struktury i unie, pracę na zmiennych i na wskaźnikach.
- Podaj tekst w komentarzach.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  struct daneos {                               //???
4      char imie[15];
5      char nazwisko[25];
6      char pesel[11];
7  };
8  union dane {                                   //???
9      struct daneos osoba;
10     char firma[60];
11 };
12 struct daneklienta {                          //???
13     int typ; //0-osoba,  inne-firma
14     union dane klient;
15 };
16 struct daneklienta wczytaj(int t);             //???
17 void wyswietl(struct daneklienta kl, int t);    //???
18
19 void wczytaj2(struct daneklienta *wsk, int t);  //???
20 void wyswietl2(struct daneklienta *wsk, int t); //???
21 //=====
22 int main(int argc, char *argv[])
23 { struct daneklienta klient1, klient2;         //???
24   int wybor;
25   printf("Klient 1\n");
26   printf("Podaj typ klienta 0 - osoba,
27           inna wartość - firma\n");
28   scanf("%d", &wybor);                          //???
29   klient1=wczytaj(wybor);                        //???
30   wyswietl(klient1, wybor);                      //???
31   printf("Klient 2\n");
32   printf("Podaj typ klienta 0 - osoba,
33           inna wartość - firma\n");
34   scanf("%d", &wybor);

```



```
35  wczytaj2(&klient2, wybor);           //???
36  wyswietl2(&klient2, wybor);          //???
37  system("PAUSE");
38  return 0;
39  }
40  //=====
41  struct daneklienta wczytaj(int t)      //???
42  { struct daneklienta temp;
43    if(t==0)
44      {printf("Podaj imie ");
45       scanf("%s",temp.klient.osoba.imie); //???
46       printf("Podaj nazwisko ");
47       scanf("%s",temp.klient.osoba.nazwisko);
48       printf("Podaj pesel ");
49       scanf("%s",temp.klient.osoba.pesel);
50       return temp;                     //???
51     }
52     else
53     {
54       printf("Podaj nazwe firmy ");
55       scanf("%s",temp.klient.firma);    //???
56       return temp;                     //???
57     }
58  }
59  void wyswietl(struct daneklienta kl, int t) //???
60  {
61    if(t==0)
62      {printf("%s %s %s \n", kl.klient.osoba.imie,
63       kl.klient.osoba.nazwisko, //???
64       kl.klient.osoba.pesel);} //???
65    else
66      {printf("%s\n", kl.klient.firma);} //???
67  }
68  void wczytaj2(struct daneklienta *wsk, int t)
69  { if(t==0)
70    { printf("Podaj imie ");
71     scanf("%s",wsk->klient.osoba.imie); //???
72     printf("Podaj nazwisko ");
73     scanf("%s",wsk->klient.osoba.nazwisko); //???
74     printf("Podaj pesel ");
75     scanf("%s",wsk->klient.osoba.pesel); //???
76    }
77    else
78    { printf("Podaj nazwe firmy ");
79     scanf("%s",wsk->klient.firma); //???
80    }
81  void wyswietl2(struct daneklienta *wsk, int t) //???
82  { if(t==0)
83    {printf("%s %s %s \n",
```



```
84     wsk->klient.osoba.imie,           // ???
85     wsk->klient.osoba.nazwisko,
86     wsk->klient.osoba.pesel);}
87     else
88     {printf("%s\n", wsk->klient.firma);}
89 }
```

### **Zadania do wykonania**

#### **Zadanie 13.3. Płaca pracownika**

Oblicz płacę pracownika fizycznego. Zadeklaruj strukturę o polach: imię nazwisko, liczba godzin, stawka, premia w %.

Napisz funkcje do wczytywania danych, obliczania płacy i wyświetlania. Zastosuj przetwarzanie struktur przez zmienną i wskaźnik. Wywołaj te funkcje.

#### **Zadanie 13.4. Płace grupy pracowników**

Oblicz sumaryczną kwotę do wypłaty dla N pracowników fizycznych. Zadeklaruj tablicę struktur o polach: imię, nazwisko, liczba godzin, stawka, premia w procentach.

Wyświetl listę płac i sumaryczną kwotę pieniędzy do wypłaty.

Podaj nazwiska osób zarabiających najwięcej.

Zdefiniuj i wywołaj odpowiednie funkcje.

#### **Zadanie 13.5. Nagrody dla pracowników**

Zadeklaruj unię do przechowywania danych o nagrodach w postaci: albo kwota pieniężna, albo wycieczka w dane miejsce i w danym terminie albo list pochwalny. Dla N pracowników przypisz nagrody różnego typu.

Wyświetl przydział nagród w postaci: imię, nazwisko i dane nagrody.

Zdefiniuj i wywołaj odpowiednie funkcje.

#### **Zadanie 13.6. Średnia ocen studenta i grupy**

Zadeklaruj strukturę student o polach: imię, nazwisko, oceny (tablica 5 ocen). Dla grupy n studentów (tablica) oblicz średnią ocen każdego studenta i średnią grupy.

Zdefiniuj i wywołaj odpowiednie funkcje.

### **Zadania dodatkowe**

#### **Zadanie 13.7. Nagrody dla studentów**

Zadeklaruj strukturę do przechowywania danych o szczęśliwych numerkach wylosowanych dla studentów (imię, nazwisko, numer). Dla grupy n studentów wpisz imiona i nazwiska, wylosuj im szczęśliwe numerki.

Ustal zwycięzców gry wg jednej z zasad:

- wygrywa osoba/osoby z największym numerkiem,
- wygrywa osoba/osoby, której numer jest najbliższy średniej wylosowanych liczb.

### **Zadanie 13.8. Cukierki**

Zadeklaruj strukturę cukierek zawierającą informacje o nazwie producenta, nazwie cukierków, cenie za 1 kg, ilości w kg, dacie produkcji, dacie przydatności do spożycia. Wprowadź dane kilku takich struktur.

Wyświetl informacje o cukierkach wybranego producenta, których cena za 1 kg jest zawarta w przedziale od  $x$  do  $y$  zł.

Wyświetl informację o nazwach i ilości cukierków, którym skończył się termin przydatności do spożycia.

Zdefiniuj i wywołaj odpowiednie funkcje.

### **Zadanie 13.9. Plan zajęć**

Zadeklaruj strukturę `pozycjaPlanu` zawierającą informację o zajęciach: dzień tygodnia, godzina rozpoczęcia, godzina zakończenia, przedmiot, wykładowca, nr sali. Wprowadź dane o swoim planie zajęć.

Wyświetl informacje o: zajęciach w danym dniu tygodnia, zajęciach z danego przedmiotu.

Podaj ile razy w tygodniu rozpoczynasz zajęcia o 8 rano.

Zdefiniuj i wywołaj odpowiednie funkcje.

### **Zadanie 13.10. Ankieta**

Zadeklaruj strukturę `ankieta` zawierającą informacje: imię, nazwisko, adres (ulica, nr domu, nr mieszkania), płeć i wiek.

Określ:

- ile kobiet i ilu mężczyzn mieszka w jednym wybranym domu,
- ilu mężczyzn w wieku <18-60> lat mieszka na jednej wybranej ulicy.

Zdefiniuj i wywołaj odpowiednie funkcje.

### **Zadanie 13.11. Stypendium dla studentów**

Napisz program wyznaczania stypendium studentom na podstawie wyników sesji zgodnie z zasadami:

- jeżeli wszystkie oceny to 5 – przyznawane jest podwyższone stypendium,
- jeżeli wszystkie oceny to 5 lub 4 – przyznawane jest zwykłe stypendium,
- jeżeli wśród ocen jest 3 – stypendium się nie należy.

Program powinien wyświetlać:

- listę studentów z ocenami i ich średnią,
- listę studentów z podwyższonym stypendium,
- listę studentów ze zwykłym stypendium.

Zdefiniuj odpowiednie funkcje dla pojedynczego studenta:

- `wczytywanieDanych`,
- `wyswietlanieDanych`,
- `przyznanieStypendium` z wynikiem: normalne/ podwyższone/ brak,
- `sredniaOcen`.





Zastosuj te funkcje do grupy  $n$  studentów, by wyświetlić ww. listy. Ustal założenia programu (oceny – liczby całkowite lub rzeczywiste, liczba ocen w sesji określana w programie lub stała).



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz Społeczny



## LABORATORIUM 14. ZŁOŻONE TYPY DANYCH. PLIKI.

### Cel laboratorium:

Zaznajomienie z obsługą plików do przechowywania danych w zewnętrznej pamięci. Nabycie praktycznych umiejętności pracy z plikami tekstowymi i binarnymi.

### Zakres tematyczny zajęć:

- *pojęcie pliku,*
- *rodzaje plików,*
- *algorytm przetwarzania plików,*
- *funkcje otwarcia i zamknięcia pliku,*
- *zapis do pliku i odczyt z pliku.*

### Kompendium wiedzy:

**Plik** to wydzielony fragment pamięci (najczęściej dyskowej), posiadający nazwę (ciąg bajtów).

#### Rodzaje plików:

- **Tekstowe** – dane w postaci znakowej (konwersja liczb na znaki),
- **Binarne** – dane w postaci wewnętrznej reprezentacji (brak konwersji, dokładne dane).

#### Obsługa plików:

- Niskopoziomowa – obsługa poprzez funkcje:  
`read()`, `open()`, `write()`, `close()` – `<io.h>`
- Wysokopoziomowa – obsługa poprzez funkcje:  
`fopen()`, `fclose()`, `fread()`, `fprintf()`, ... – `<stdio.h>`

#### Przetwarzanie plików metodą wysokopoziomową:

1. Otwarcie pliku: `fopen()`.
2. Wykonanie operacji na pliku (zapis, odczyt, szukanie, obliczanie, ...).
  - Funkcje zapisu:  
`putc()`, `fputs()`, `fprintf()`, `fwrite()`
  - Funkcje odczytu:  
`getc()`, `fgets()`, `fscanf()`, `fread()`
  - Funkcje pomocnicze:  
`feof()`, `fseek()`, `rewind()`, `ftell()`, ...
3. Zamknięcie pliku: `fclose()`.

**Otwarcie pliku** – zwraca wskaźnik na strukturę typu `FILE` (identyfikator pliku) lub wskaźnik zerowy (`NULL`).

**Zamknięcie pliku** – zwraca 0, jeśli operacja się powiodła, lub `EOF`, jeśli nie.

```
fopen(nazwa_pliku, tryb_otwarcia); //otwarcie
fclose(wskaźnik_pliku); //zamknięcie
```



**Tryb otwarcia dla plików tekstowych:**

"r" – odczyt,  
"w" – zapis (nadpisywanie lub tworzenie),  
"a" – zapis na końcu (dopisywanie lub tworzenie),  
"r+" – odczyt i zapis,  
"w+" – odczyt i zapis (nadpisywanie lub tworzenie),  
"a+" – odczyt i zapis na końcu (dopisywanie lub tworzenie).

**Tryb otwarcia dla plików binarnych:**

"rb"; "wb"; "ab" ; "rb+"; "r+b"; "wb+" ; "w+b" ;  
"ab+"; "a+b"

```
FILE *fp;  
fp=fopen("test.txt", "w");  
fclose(fp);
```

**Zapis do pliku:**

```
putc(znak, wskaźnik_pliku);  
fputs(*tekst, wskaźnik_pliku);  
fprintf(wskaźnik_pliku, format, dane);  
fwrite(adres_w_pamięci, rozmiar_bloku, ilość_bloków,  
wskaźnik_pliku);
```

```
FILE *fp; char ch; char * slowo; int dane;  
putc(ch,fp); //zapis znaku  
fputs(slowo,fp); // zapis łańcucha znaków  
fprintf(fp,"%d",dane); //zapis z formatem
```

**Odczyt z pliku:**

```
getc(wskaźnik_pliku);  
fgets(*tekst, dlugosc, wskaźnik_pliku);  
fscanf(wskaźnik_pliku, format, dane);  
fread(adres_w_pamięci, rozmiar_bloku, ilość_bloków,  
wskaźnik_pliku);  
ilość_bloków, wskaźnik_pliku);
```

```
FILE *fp; char ch; char buf[256]; int dane;  
...  
ch= getc(fp); //odczyt znaku  
fgets(buf, 256,fp); // odczyt łańcucha znaków  
fscanf(fp,"%d",dane); //odczyt z formatem
```

**Funkcje pomocnicze:**

- Przesuwanie wskaźnika w pliku:

```
int fseek(wskaźnik_pliku, pozycja, tryb);
```

gdzie: tryb – sposób liczenia pozycji/przesunięcia (0/ 1/ 2),  
pozycja – pozycja wskaźnika w pliku uzależniona od trybu:



Jeśli tryb=SEEK\_SET (0) => pozycja liczona jest od początku.

- o Jeśli tryb=SEEK\_CUR (1) => pozycja jest liczona jako przesunięcie od aktualnej pozycji,
- o Jeśli tryb=SEEK\_END (2) => pozycja jest liczona jako przesunięcie od końca pliku (wskaźnik pliku jest przesuwany do pozycji <EOF> + pozycja).

```
FILE *fp;
```

```
...
```

```
fseek(fp, 10, 0); // dziesiąta pozycja, licząc od początku
```

```
fseek(fp, 10, 1); // dziesiąta pozycja, licząc od bieżącej
```

```
fseek(fp, -10, 2); // dziesiąta pozycja, licząc od końca pliku
```

- Ustawienie wskaźnika na początek pliku:

```
rewind(wskaźnik pliku);
```

```
...
```

```
rewind(fp);
```

```
...
```

- Pobranie aktualnej pozycji z pliku:

```
ftell(wskaźnik pliku);
```

```
...
```

```
fseek (fp, 0, SEEK_END);
```

```
long size=ftell (pFile);
```

```
...
```

- Informowanie o osiągnięciu/nie osiągnięciu pozycji końca pliku (prawda – koniec pliku, fałsz – brak końca pliku:

```
feof(wskaźnik pliku);
```

```
...
```

```
while(!feof(fp)) { //odczyt }
```

```
...
```

### Pytania kontrolne:

1. Co to jest plik?
2. Jakie są rodzaje plików?
3. Podaj algorytm przetwarzania plików metodą wyskokopoziomową.
4. Podaj funkcje otwarcia i zamknięcia pliku.
5. Podaj funkcje zapisu danych do pliku.
6. Podaj funkcje odczytu danych z pliku.
7. Jak przesuwac wskaźnik w pliku?
8. Jak uzyskać informację gdzie znajduje się wskaźnik w pliku?

### Zadania do analizy

#### Zadanie 14.1. Imiona przyjaciół w pliku tekstowym

- Przeanalizuj przykład programu wykorzystującego pliki tekstowe.
- Podaj tekst w komentarzach.



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int zapisT1(char nazwa[20], FILE *fpplik); //???
5  int odczytT1(char nazwa[20], FILE *fpplik); //???
6  //=====
7  int main(int argc, char *argv[])
8  { char nazwa1[]="lista.txt"; //???
9    FILE *f; //???
10   int wynik;
11   wynik=zapisT1(nazwa1,f); //???
12   if(wynik==0)printf("Operacja zapisu ok\n"); //???
13   wynik=odczytT1(nazwa1,f); //???
14   if(wynik==0)printf("Operacja odczytu ok\n"); //???
15   system("PAUSE");
16   return EXIT_SUCCESS;
17 }
18 //=====
19 int zapisT1(char nazwa[20], FILE *fpplik) //???
20 {char nazwisko[25]; int i=0;
21 if((fpplik=fopen(nazwa, "a"))==NULL) //???
22     {printf("Blad otwarcia\n");
23     system("PAUSE"); abort();}
24 printf("Podaj nazwiska konczac enterem\n");
25 while(gets(nazwisko)!=NULL && nazwisko[0]!='\0') //???
26 {fprintf(fpplik, "%s\n", nazwisko);i++; //???
27 }
28 if(fclose(fpplik)!=0){exit(2);} //???
29 printf("Do pliku zapisano %d nazwisk\n",i);
30 return 0;
31 }
32 //-----
33 int odczytT1(char nazwa[20], FILE *fpplik) //???
34 {char nazwisko[25]; int i=0;
35 if((fpplik=fopen(nazwa, "r"))==NULL) //???
36     {printf("blad otwarcia\n");
37     system("PAUSE"); abort();}
38 printf("\nZawartosc pliku %s\n",nazwa);
39 while(fscanf(fpplik,"%s",nazwisko)==1) //???
40 {puts(nazwisko);i++; //???
41 }
42 if(fclose(fpplik)!=0){exit(2);} //???
43 printf("\nZ pliku odczytano %d nazwisk\n",i);
44 return 0;
45 }
```



**Zadanie 14.2. Dane studenta w pliku binarnym**

- Przeanalizuj przykład programu wykorzystującego pliki binarne.
- Podaj tekst w komentarzach.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int zapisB(char nazwa[20], FILE *fpplik);           //???
5  int odczytB(char nazwa[20], FILE *fpplik);         //???
6  //=====
7  int main(int argc, char *argv[])
8  { char nazwa2[]="studenci.txt";
9    FILE *f;
10   int wynik;
11   wynik=zapisB(nazwa2,f);                           //???
12   if(wynik==0)printf("operacja zapisu ok\n");        //???
13   wynik=odczytB(nazwa2,f);                           //???
14   if(wynik==0)printf("operacja odczytu ok\n");       //???
15   system("PAUSE");
16   return EXIT_SUCCESS;
17 }
18 //=====
19 int zapisB(char nazwa[20], FILE *fpplik)             //???
20 {struct student st; int i;                           //???
21   int rozmiar=sizeof(struct student);                //???
22   int licznik=1; //liczba zapisów
23   if ((fpplik=fopen(nazwa, "ab"))==NULL)             //???
24     { printf ("błąd");exit(1); }
25   printf("Podaj liczbę zapisów ");
26   scanf("%d", &licznik);
27   for(i=1;i<=licznik;i++)
28     {printf("Podaj nazwisko %d: ",i);
29       scanf("%s", st.nazwisko);                      //???
30       printf("Podaj ocene %d: ",i);
31       scanf("%d", &st.ocena);                        //???
32       fwrite(&st, rozmiar, 1, fpplik);               //???
33     }
34   if (fclose(fpplik) !=0) {printf ("Bład "); }        //???
35   return 0;
36 }
37 //-----
38 int odczytB(char nazwa[20], FILE *fpplik)            //???
39 { struct student st;                                  //???
40   int rozmiar=sizeof(struct student);                //???
41   int licznik=0;
42   if ((fpplik=fopen(nazwa, "rb"))==NULL)             //???
43     { printf ("błąd");exit(1); }
44   printf("Zawartość pliku %s\n", nazwa);

```



```
45 while(fread(&st,rozmiar,1,fplik)==1) //???
46     {printf("student: %s ocena: %d\n",
47         st.nazwisko,st.ocena); //???
48         licznik++;}
49     printf("liczba pozycji: %d\n",licznik);
50     if (fclose(fplik) !=0) //???
51         {printf ("Blad przy zamykaniu pliku"); }
52 return 0;
53 }
```

### **Zadania do wykonania**

#### **Zadanie 14.3. Płaca pracownika w pliku tekstowym**

Oblicz płacę pracownika fizycznego i zapisz ją do pliku tekstowego. Zadeklaruj strukturę o polach: imię, nazwisko, liczba godzin, stawka, premia w % i do wypłaty (pole wyliczane). Napisz funkcje zapisu i odczytu z pliku. Wywołaj te funkcje kilkakrotnie. Odczytaj plik.

#### **Zadanie 14.4. Płaca pracownika w pliku binarnym**

Oblicz płacę pracownika fizycznego i zapisz ją do pliku binarnego. Zadeklaruj strukturę o polach: imię, nazwisko, liczba godzin, stawka, premia w % i do wypłaty (pole wyliczane). Napisz funkcje zapisu i odczytu z pliku. Wywołaj te funkcje kilkakrotnie. Odczytaj plik. Wyświetl dane pracowników, których kwota do wypłaty przekracza podaną wartość.

#### **Zadanie 14.5. Baza danych książek**

Załącz plik, będący prostą kartotekową bazą danych książek. Zadeklaruj strukturę opisującą pozycję bibliograficzną. Napisz funkcje zapisu i odczytu z pliku. Zapis pozycji (liczba zapisów nie jest określona) zakończ umownym znakiem (np. \* zamiast nazwiska autora). Napisz funkcję wyświetlającą tytuły książek podanego autora.

#### **Zadanie 14.6. Pomiary temperatur w pliku**

Załącz plik z pomiarami temperatur (liczby rzeczywiste). Napisz funkcje zapisu i odczytu z pliku. Wywołaj te funkcje. Odczytaj plik. Oblicz średnią arytmetyczną z pomiarów przechowywanych w pliku.

### **Zadania dodatkowe**

#### **Zadanie 14.7. Dostęp swobodny do elementu pliku**

Załącz plik z n wylosowanymi liczbami całkowitymi. Napisz funkcje zapisu i odczytu z pliku. Wywołaj te funkcje. Odczytaj plik. Wyświetl element pliku na podanej pozycji.

Wskazówka: wykorzystaj funkcję `fseek()`.



### Zadanie 14.8. Zawody sportowe

Załącz plik z wynikami zawodów sportowych. Nazwa pliku to nazwa konkurencji. Zawartość pliku to: imię, nazwisko i wynik zawodnika. Napisz funkcje zapisu i odczytu z pliku. Wywołaj te funkcje. Odczytaj plik. Wyświetl trzy najlepsze wyniki i dane zawodników, którzy je otrzymali.

### Zadanie 14.9. Eksport towarów

Załącz plik z danymi o eksportowanych za granicę towarach zawierający dane: nazwa towaru, kraj eksportu i wielkość eksportu w sztukach. Wyświetlić listę krajów, które eksportują podany towar i podać ogólną wielkość importu.

Napisz program, w którym wczytane są dane, wywołane odpowiednie funkcje, wyświetlone wyniki.

### Zadanie 14.10. Kopiowanie plików

Napisz program kopiujący co trzeci znak z jednego pliku do drugiego. Wyświetl zawartość obydwu plików.

### Zadanie 14.11. Pobieranie danych o plikach z argumentów wiersza poleceń

Napisz program, który do pliku o podanej nazwie wpisuje  $n$  kolejnych liczb całkowitych (od 1), a następnie wyświetla zawartość podanego pliku i średnią arytmetyczną liczb z pliku. Informację o nazwie pliku i informację ile liczb należy zapisać w pliku program pobiera z linii poleceń. W przypadku braku argumentów linii poleceń nazwa pliku to `dane.txt`, a  $n=10$ .

#### Wskazówka:

Funkcja `main` posiada dwa argumenty:

- argument `argc` (typu całkowitego) – przechowuje liczbę słów wpisanych w linii poleceń, uwzględniając nazwę programu,
- tablica łańcuchów `argv[]`, przechowuje słowa wpisane w linii poleceń:
  - `argv[0]` jest nazwą programu,
  - `argv[1]` jest pierwszym argumentem wywołanego programu (tylko wtedy, gdy w linii poleceń wpisujemy coś więcej niż tylko nazwę programu),
  - `argv[2]` – kolejnym argumentem, itd.
- wykorzystaj funkcję konwersji tekstu na liczbę.

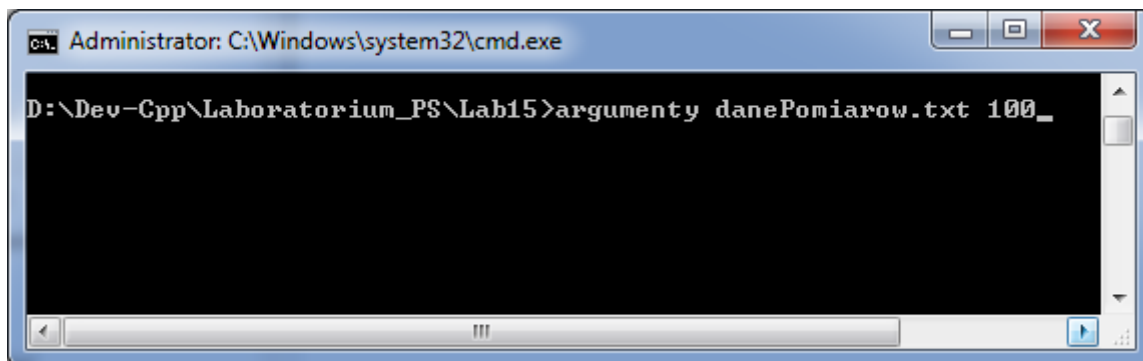
Poniższy program wyświetla wszystkie argumenty podane w linii poleceń:

```
1 #include <stdio.h>
2
3 int main(int argc, char *argv[])
4 {
5     int i;
6     for(i=0;i<argc;i++)
7         printf("%s\n",argv[i]);
8     return 0;
9 }
```

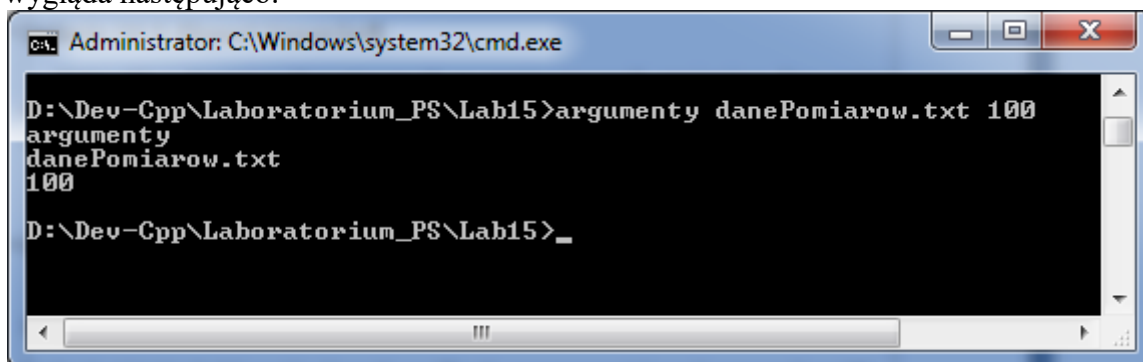




Przykładowe działanie programu, o nazwie `argumenty.exe`, uruchomionego z wiersza poleceń w oknie konsoli `cmd`:



wygląda następująco:



Interpretacja wyniku:

`Argc =3`

`Argv[0]` – `argumenty.exe` – nazwa uruchamianego pliku

`Argv[1]` – `danePomiarow.txt`

`Argv[2]` – `100`

## **LABORATORIUM 15. KOŁOKWIUM 2.**

### **Cel laboratorium:**

Weryfikacja nabytych umiejętności pisania programów z wykorzystaniem instrukcji sterujących i zmiennych typu złożonego (tablic, struktur, unii i plików).

### **Wytyczne do kolokwium 2:**

- zakres laboratoriów 1-14,
- próg zaliczeniowy 60%,
- pozostałe wytyczne i sposób zaliczenia kolokwium ustala prowadzący zajęcia.



**Fundusze Europejskie**  
Wiedza Edukacja Rozwój



**Rzeczpospolita  
Polska**

**Unia Europejska**  
Europejski Fundusz Społeczny





Materiały zostały opracowane w ramach projektu  
„*Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga*”,  
umowa nr **POWR.03.05.00-00-Z060/18-00**  
w ramach Programu Operacyjnego Wiedza Edukacja Rozwój 2014-2020  
współfinansowanego ze środków Europejskiego Funduszu Społecznego