

# Podstawy Programowania

## Wykład nr 9: Wejście/wyjście

dr hab. inż. Dariusz Dereniowski

Katedra Algorytmów i Modelowania Systemów  
Wydział ETI, Politechnika Gdańska

## Wejście i wyjście w C

- `#include <stdio.h>` pozwala korzystać z funkcji standardowego wejścia i wyjścia w języku C
- funkcje wejścia/wyjścia zwracają wartości, które pozwalają na wykrycie ewentualnych sytuacji błędnych/szczególnych
- sytuacje takie to np.:
  - brak danych przy próbie czytania z wejścia
  - zły format danych wejściowych
  - błąd przy wypisywaniu danych na wyjście

## Standardowe wyjście

```
int putchar( int c );
```

Funkcja wypisuje znak o kodzie ASCII `c` na standardowe wyjście i zwraca kod ASCII wypisanego znaku jako `unsigned char` rzutowany na `int`, oraz zwraca stałą `EOF` w przypadku błędu.

```
int printf( const char *format, ... );
```

- Funkcja `printf` wypisuje na standardowe wyjście napis `format`, w którym wartości sekwencji kontrolnych, rozpoczynających się od znaku `%` są zastępowane wartościami zmiennych podanych jako dodatkowe argumenty w miejsce `...`
- `printf` zwraca liczbę znaków, które zostały wypisane.

## Sekwencje kontrolne

Sekwencje kontrolne:

Sekw.	Typ	Co wypisuje?
%d	int	wartość dziesiętną
%c	char	znak o zadanym kodzie ASCII
%s	char *	napis
%p	void *	adres
%f	float	wartość liczby zmiennoprzec.
%e	float	j.w., lecz notacja wykładnicza

Modyfikatory:

Mod.	Działanie.
l	"long"
0	uzup. puste pola zerami
liczba	określa szerokość pola
.liczba	liczba miejsc po przec.

```
#include <stdio.h>
#include <math.h>
```

```
int main() {
    double pi = acos(-1.0), potega;
    int i;
    char c = 'a';
    long l = 10;
    long long ll = 20;
```

```
    for ( i=0, potega=pi; i < 5; i++, potega *= pi )
        printf( "%.4f do potęgi %2d = %.6le\n", (float)pi, i+1, potega );
```

```
    printf( "Kod ASCII litery %c wynosi %d\n", c, (int)c );
    printf( "Liczba typu long: %ld,\n... oraz typu long long: %lld\n", l, ll );
    printf( "Adres zmiennej l to %p\n", (void *)&l );
    return 0;
}
```

```
3.1416 do potęgi 1 = 3.141593e+00
3.1416 do potęgi 2 = 9.869604e+00
3.1416 do potęgi 3 = 3.100628e+01
3.1416 do potęgi 4 = 9.740909e+01
3.1416 do potęgi 5 = 3.060197e+02
Kod ASCII litery a wynosi 97
Liczba typu long: 10,
... oraz typu long long: 20
Adres zmiennej l to 0xbfd1134
```

## Standardowe wejście

```
int getchar( void );
```

Funkcja pobiera jeden znak ze standardowego wejścia i zwraca ten znak jako `unsigned char` rzutowany na `int`, oraz zwraca stałą `EOF` w przypadku błędu lub końca pliku (końca wejścia)

```
int scanf( const char *format, ... );
```

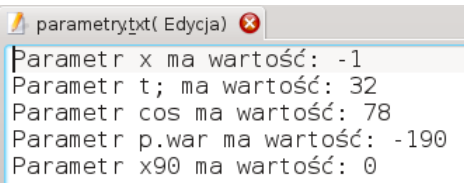
- Funkcja `scanf`, zgodnie z formatem podanym w napisie `format`, dopasowuje ciąg znaków ze standardowego wejścia, modyfikując wartości podanych zmiennych.
- `scanf` zwraca liczbę dopasowanych parametrów.
- Uwaga: parametry `...` to adresy zmiennych.

## scanf – przykład

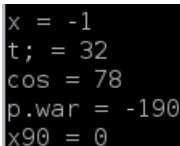
Mamy dany plik o nazwie `parametry.txt`, którego każda linia ma postać: "Parametr ... ma wartość: ...", gdzie w miejscu pierwszego "..." mamy dowolne słowo (bez białych znaków) natomiast w miejscu drugiego "..." mamy liczbę całkowitą. Interesuje nas odcodowanie nazw wszystkich parametrów oraz ich wartości. Plik ten zostanie przekierowany na standardowe wejście, tzn program o nazwie `prog.exe` uruchomimy następująco: `prog.exe <parametry.txt`.

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

```
int main() {  
    char parametr[128];  
    int wartosc;  
  
    while ( scanf( "Parametr %s ma wartość: %i\n", parametr, &wartosc ) > 0 )  
        printf( "%s = %d\n", parametr, wartosc );  
    return 0;  
}
```



```
parametry.txt( Edycja) x  
Parametr x ma wartość: -1  
Parametr t; ma wartość: 32  
Parametr cos ma wartość: 78  
Parametr p.war ma wartość: -190  
Parametr x90 ma wartość: 0
```



```
x = -1  
t; = 32  
cos = 78  
p.war = -190  
x90 = 0
```

## Otwieranie i zamykanie plików

```
FILE *fopen( const char *sciezka, const char *tryb );
```

- Typ **FILE \*** przechowuje wskaźnik na specjalny typ opisujący plik.
- Funkcja zwróci **NULL** jeśli otwarcie pliku nie powiodło się.
- Napis **tryb** zaczyna się od jednej z następujących sekwencji kontrolnych:
  - **r** – otwarcie pliku do odczytu.
  - **r+** – otwarcie pliku do odczytu i zapisu.
  - **w** – utwórz pusty plik do zapisu; jeśli plik istnieje, to jest nadpisany pustym.
  - **w+** – otwarcie pliku do zapisu i odczytu. Jeśli plik istnieje, to jest nadpisany pustym. We wszystkich dotychczasowych przypadkach, wskaźnik odczytu/zapisu do pliku jest inicjalnie ustawiany na jego początek.
  - **a** – otwarcie pliku w celu dopisywania na jego końcu. Plik jest tworzony jeśli nie istnieje.
  - **a+** – otwarcie pliku do odczytu i dopisywania na jego końcu. Wskaźnik odczytu jest ustawiany na początek pliku, lecz zapisywanie do pliku odbywa się na jego końcu.
- Ustawienie wskaźnika odczytu/zapisu pliku można dokonać funkcją **fseek**.
- Plik zamykamy używając funkcji **int fclose( FILE \* );**. (Zwraca **EOF** w przypadku błędu.)

## Zapisywanie/odczyt do/z pliku (tryb tekstowy)

Do zapisywania (formatowanego/tekstowego) do pliku służą m.in. funkcje:

```
int fprintf( FILE *f, const char *format, ... );  
int fputc( int c, FILE *f);  
int fputs( const char *napis, FILE *f);
```

Do odczytywania (formatowanego/tekstowego) z pliku służą m.in. funkcje:

```
int fscanf( FILE *f, const char *format, ... );  
int fgetc( FILE *f);  
int fgets( const char *napis, int rozmiar, FILE *f);
```

Przy używaniu powyższych funkcji obowiązują analogiczne zasady jak przy ich odpowiednikach działających na standardowym wejściu/wyjściu, z wyjątkiem dodatkowego parametru wskazującego na plik.



## Zapisywanie/odczyt do/z pliku (tryb tekstowy) – przykład

```
#include <stdio.h>
#define LPOTEG 20
/* program wpisuje kolejne potęgi liczby 2 do pliku potegi.txt.
   Kolejne liczby są oddzielone w pliku znakiem nowej linii */
int main() {
    FILE *plik;
    unsigned long long pot2 = 1;

    plik = fopen( "potegi.txt", "w" );
    if ( plik == NULL ) {
        printf( "Otwarcie pliku nie powiodło się\n" );
        return 1;
    }

    for ( int i=0; i < LPOTEG; i++ )
        fprintf( plik, "%lld\n", pot2<<i );

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

```
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
```

## Zapisywanie/odczyt do/z pliku (tryb binarny)

Do zapisywania (binarnego) do pliku służy funkcja:

```
size_t fwrite( const void *w, size_t rozmiar, size_t l_el, FILE *f);
```

 Do

odczytywania (binarnego) z pliku służy funkcja:

```
size_t fread( void *w, size_t rozmiar, size_t l_el, FILE *f);
```

- Funkcje zapisują/odczytują `l_el` elementów, każdy rozmiaru `rozmiar`, znajdujących się w pamięci pod adresem `w`.
- Obie funkcje zwracają liczbę odczytanych lub zapisanych elementów (spośród `l_el` wskazanych do zapisu/odczytu). Aby otrzymać łączną liczbę zapisanych/odczytanych bajtów, należy zwróconą wartość wymnożyć przez wartość parametru `rozmiar`.

## Przykład: zapis binarny tablicy liczb do pliku

```
#include <stdio.h>
#define MAX 20
/* program wpisuje kolejne potęgi liczby 2 do pliku potegi.txt.
   Kolejne liczby są oddzielone w pliku znakiem nowej linii */
int main() {
    FILE *plik;
    int tab[MAX], i;

    /* wypełniamy tablicę kolejnymi liczbami naturalnymi */
    for ( i=0; i < MAX; i++ )
        tab[i] = i;

    plik = fopen( "liczby.bin", "w" );
    if ( plik == NULL ) {
        printf( "Otwarcie pliku nie powiodło się\n" );
        return 1;
    }

    fwrite( (void *)tab, sizeof tab[0], MAX, plik );
    fclose( plik );
    return 0;
}
```

- Plik `liczby.bin` będzie miał dokładnie  $\text{MAX} * (\text{sizeof tab}[0])$  bajtów, bez względu na zawartość tablicy `tab`.
- *i*-ta liczba z tablicy `tab` (traktując liczbę pod indeksem 0 jako pierwszą) jest zapisana na kolejnych `sizeof(int)` bajtach, począwszy od bajtu numer  $(i-1) * \text{sizeof(int)}$ .