

# Informatyka 2 (ES1E3017)

Politechnika Białostocka - Wydział Elektryczny  
Elektrotechnika, semestr III, studia stacjonarne I stopnia  
Rok akademicki 2020/2021

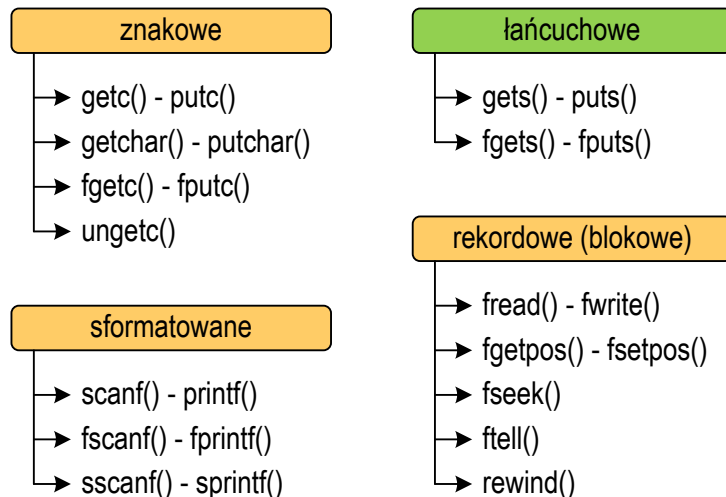
## Wykład nr 5 (03.11.2020)

dr inż. Jarosław Forenc

## Plan wykładu nr 5

- Typy operacji wejścia-wyjścia
  - łańcuchowe
  - sformatowane
  - rekordowe (blokowe)

## Łańcuchowe operacje wejścia-wyjścia



## Łańcuchowe operacje wejścia-wyjścia

```
GETS stdio.h  
char* gets(char *buf);
```

- Pobiera do bufora pamięci wskazywanego przez argument **buf** linię znaków ze strumienia **stdin** (standardowo klawiatura)
- Wczytywanie jest kończone po napotkaniu znacznika nowej linii **'\n'**, który zastępowany jest znakiem końca łańcucha **'\0'**
- Funkcja **gets()** umożliwi wczytanie łańcucha znaków zawierającego spacje i tabulatory
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca wskazanie do łańcucha **buf**
- Jeśli wystąpił błąd lub podczas wczytywania został napotkany znacznik końca pliku, to funkcja zwraca wartość **EOF**

## Łańcuchowe operacje wejścia-wyjścia

**PUTS** stdio.h

```
int puts(const char *buf);
```

- Wpisuje łańcuch `buf` do strumienia `stdout` (standardowo ekran), zastępując znak `'\0'` znakiem `'\n'`
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca ostatni wypisany znak
- Jeśli wystąpił błąd, to funkcja zwraca wartość `EOF`

```
char tablica[80];  
  
gets(tablica);  
puts(tablica);
```

## Łańcuchowe operacje wejścia-wyjścia

**FGETS** stdio.h

```
char* fgets(char *buf, int max, FILE *fp);
```

- Pobiera znaki z otwartego strumienia reprezentowanego przez `fp` i zapisuje je do bufora pamięci wskazanego przez `buf`
- Pobieranie znaków jest przerywane po napotkaniu znacznika końca linii `'\n'` lub odczytaniu `max-1` znaków
- Po ostatnim przeczytanym znaku wstawia do bufora `buf` znak `'\0'`
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca wskazanie do łańcucha `buf`
- Jeśli wystąpił błąd lub napotkano znacznik końca pliku, to funkcja zwraca wartość `NULL`

## Łańcuchowe operacje wejścia-wyjścia

**FPUTS** stdio.h

```
int fputs(const char *buf, FILE *fp);
```

- Wpisuje łańcuch `buf` do strumienia `fp`, nie dołącza znaku końca wiersza `'\n'`
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca ostatni wypisany znak
- Jeśli wystąpił błąd, to funkcja zwraca wartość `EOF`

## Przykład: wyświetlenie pliku tekstowego

```
#include <stdio.h>  
  
int main(void)  
{  
    FILE *fp;  
    char buf[15];  
  
    fp = fopen("test.txt", "r");  
  
    while (fgets(buf, 15, fp) != NULL)  
        fputs(buf, stdout);  
  
    fclose(fp);  
  
    return 0;  
}
```

## Przykład: wyświetlenie pliku tekstowego

- Zawartość pliku `test.txt`

```
Poprzednikiem języka C \r \n  
był język B, \r \n  
który \r \n  
Ritchie rozwinał w język C. \r \n
```

- Kolejne wywołania funkcji `fgets(buf,15,fp);`

```
Poprzednikiem języka C \r \n  
był język B, \r \n  
który \r \n  
Ritchie rozwinał w język C. \r \n
```

## Przykład: wyświetlenie pliku tekstowego

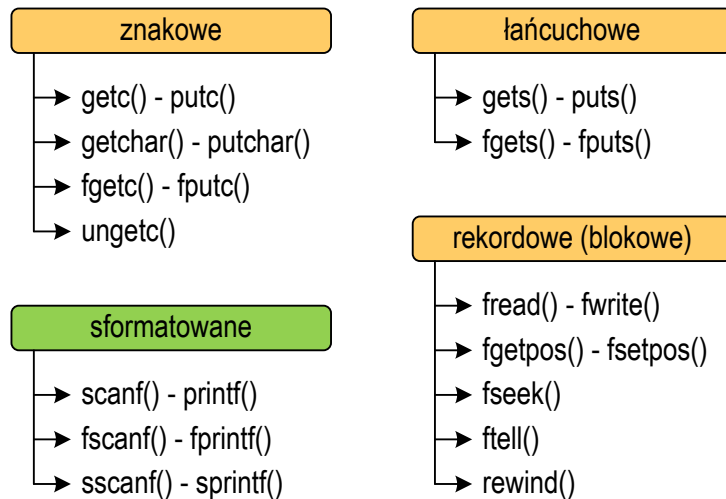
- Kolejne wywołania funkcji `fgets(buf,15,fp);`; i zawartość tablicy `buf`

```
Poprzednikiem języka C \r \n  
był język B, \r \n  
który \r \n  
Ritchie rozwinał w język C. \r \n
```

```
P o p r z e d n i k i e m \0  
j e z y k a   C \r \n  
b y l   j e z y k   B , \r \n  
k t o r y \r \n  
R i t c h i e   r o z w i n \0  
a l   w   j e z y k   C . \r \n
```

`\r \n` = `\n`

## Sformatowane operacje wejścia-wyjścia



## Sformatowane operacje wejścia-wyjścia

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

- Czyta dane ze strumienia `stdin` (klawiatura)

**FSCANF** stdio.h  
`int fscanf(FILE *fp, const char *format, ...);`

- Czyta dane z otwartego strumienia (pliku) `fp`

**SSCANF** stdio.h  
`int sscanf(char *buf, const char *format, ...);`

- Czyta dane z bufora pamięci wskazywanego przez `buf`

## Sformatowane operacje wejścia-wyjścia

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

- Wyprowadza dane do strumienia `stdout` (ekran)

**FPRINTF** stdio.h  
`int fprintf(FILE *fp, const char *format, ...);`

- Wyprowadza dane do otwartego strumienia (pliku) `fp`

**SPRINTF** stdio.h  
`int sprintf(char *buf, const char *format, ...);`

- Wyprowadza dane do bufora pamięci wskazywanego przez `buf`

## Przykład: zapisanie liczb do pliku tekstowego

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

int main(void)
{
    FILE *fp; float x; int i;
    srand((unsigned int)time(NULL));
    fp = fopen("liczby.txt", "w");
    for (i=0; i<10; i++)
    {
        x = (float)rand()/RAND_MAX*100;
        fprintf(fp, "%f\n", x);
    }
    fclose(fp);
    return 0;
}
```

```
3.830073
70.848717
99.322487
19.812616
7.132175
49.134800
10.238960
18.668173
8.914456
69.258705
```

## Przykład: zapisanie danych do pliku tekstowego

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int wiek = 21;
    float wzrost = 1.78f;
    char imie[10] = "Jan", nazw[10] = "Kowalski";

    fp = fopen("dane.txt", "w");
    fprintf(fp, "Imie: %s\n", imie);
    fprintf(fp, "Nazwisko: %s\n", nazw);
    fprintf(fp, "Wiek: %d [lat]\n", wiek);
    fprintf(fp, "Wzrost: %.2f [m]\n", wzrost);
    fclose(fp);

    return 0;
}
```

```
Imie: Jan
Nazwisko: Kowalski
Wiek: 21 [lat]
Wzrost: 1.78 [m]
```

## Obsługa błędów wejścia-wyjścia

**FEOF** stdio.h  
`int feof(FILE *fp);`

- Sprawdza, czy podczas ostatniej operacji wejścia dotyczącej strumienia `fp` został osiągnięty koniec pliku
- Zwraca wartość różną od zera, jeśli podczas ostatniej operacji wejścia został wykryty koniec pliku, w przeciwnym razie zwraca wartość 0 (zero)

## Przykład: odczytanie liczb z pliku tekstowego

```
#include <stdio.h>
int main(void)
{
    FILE *fp; float x;
    fp = fopen("liczby.txt", "r");
    fscanf(fp, "%f", &x);
    while(!feof(fp))
    {
        printf("%f\n", x);
        fscanf(fp, "%f", &x);
    }
    fclose(fp);
    return 0;
}
```

```
3.830073
70.848717
99.322487
19.812616
7.132175
49.134800
10.238960
18.668173
8.914456
69.258705
```

## Przykład: odczytanie liczb z pliku tekstowego

- Sposób zapisu liczb w pliku wejściowym nie ma znaczenia dla prawidłowości ich odczytu
- Liczby powinny być oddzielone od siebie znakami spacji, tabulacji lub znakiem nowego wiersza

```
3.830073
70.848717
99.322487
19.812616
7.132175
49.134800
10.238960
18.668173
8.914456
69.258705
```

```
3.830073    70.848717
99.322487   19.812616
7.132175    49.134800
10.238960   18.668173
8.914456    69.258705
```

```
3.830073    70.848717    99.322487
19.812616   7.132175     49.134800
10.238960   18.668173    8.914456
69.258705
```

## Przykład: odczytanie danych z pliku tekstowego

- Odczytanie danych różnych typów z pliku tekstowego

```
Nowak Grzegorz 15-12-2000
Kowalski Wojciech 03-05-1997
Jankowska Anna 23-05-1995
Mazur Krzysztof 14-01-1990
Krawczyk Monika 03-11-1995
Piotrowska Maja 12-06-1998
Dudek Piotr 31-12-1996
Pawlak Julia 01-01-1997
```

```
Grzegorz    Nowak      wiek: 20
Wojciech    Kowalski   wiek: 23
Anna        Jankowska  wiek: 25
Krzysztof   Mazur      wiek: 30
Monika      Krawczyk   wiek: 25
Maja        Piotrowska wiek: 22
Piotr       Dudek      wiek: 24
Julia       Pawlak     wiek: 23
```

## Przykład: odczytanie danych z pliku tekstowego

```
#include <stdio.h>
int main()
{
    FILE *fp;
    char naz[20], im[20];
    int d, m, r;

    fp = fopen("osoby.txt", "r");
    fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    while(!feof(fp))
    {
        printf("%-12s %-12s wiek: %d\n", im, naz, 2020-r);
        fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    }
    fclose(fp);
    return 0;
}
```

## Przykład: odczytanie danych z pliku tekstowego

```
#include <stdio.h>

int main()
{
    FILE *fp;
    char naz[20], im[20];
    int d, m, r;

    fp = fopen("osoby.txt", "r");
    fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    while(!feof(fp))
    {
        printf("%-12s %-12s wiek: %d\n", im, naz, 2020-r);
        fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);
    }
    fclose(fp);
    return 0;
}
```

Grzegorz	Nowak	wiek: 20
Wojciech	Kowalski	wiek: 23
Anna	Jankowska	wiek: 25
Krzysztof	Mazur	wiek: 30
Monika	Krawczyk	wiek: 25
Maja	Piotrowska	wiek: 22
Piotr	Dudek	wiek: 24
Julia	Pawlak	wiek: 23

## Rekordowe (blokowe) operacje wejścia-wyjścia

### znakowe

- getc() - putc()
- getchar() - putchar()
- fgetc() - fputc()
- ungetc()

### łańcuchowe

- gets() - puts()
- fgets() - fputs()

### rekordowe (blokowe)

- fread() - fwrite()
- fgetpos() - fsetpos()
- fseek()
- ftell()
- rewind()

### sformatowane

- scanf() - printf()
- fscanf() - fprintf()
- sscanf() - sprintf()

## Rekordowe (blokowe) operacje wejścia-wyjścia

**FWRITE** stdio.h

```
size_t fwrite(const void *p, size_t s, size_t n, FILE *fp);
```

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

## Przykład: zapisanie danych do pliku binarnego

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int x = 10, tab[5] = {1,2,3,4,5};
    float y = 1.2345f;

    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);
    fclose(fp);

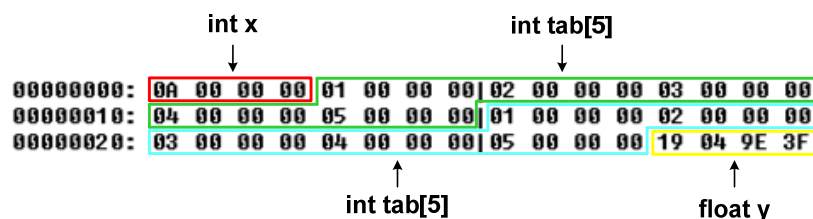
    return 0;
}
```

## Przykład: zapisanie danych do pliku binarnego

- Czterokrotne wywołanie funkcji `fwrite()`

```
fwrite(&x, sizeof(int), 1, fp); // int x = 10;
fwrite(tab, sizeof(int), 5, fp); // int tab[5] = {1,2,3,4,5};
fwrite(tab, sizeof(tab), 1, fp); // int tab[5] = {1,2,3,4,5};
fwrite(&y, sizeof(float), 1, fp); // float y = 1.2345;
```

spowoduje zapisanie do pliku 48 bajtów:



## Rekordowe (blokowe) operacje wejścia-wyjścia

```
FREAD stdio.h
size_t fread(void *p, size_t s, size_t n,
             FILE *fp);
```

- Pobiera `n` elementów o rozmiarze `s` bajtów każdy, z pliku wskazywanego przez `fp` i umieszcza odczytane dane w obszarze pamięci wskazywanym przez `p`
- 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`)

## Przykład: odczytanie liczb 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))
    {
        ile++; printf("%d\n", x);
        fread(&x, sizeof(int), 1, fp);
    }
    fclose(fp);
    printf("Odczytano: %d liczb\n", ile);
    return 0;
}
```

```
37
31
83
27
6
62
31
50
Odczytano: 8 liczb
```

## Przykład: odczytanie liczb z pliku binarnego

- Po otwarciu pliku wskaźnik pozycji pliku pokazuje na jego początek

```
↓
25 00 00 00 1F 00 00 00|53 00 00 00 1B 00 00 00 | %S
06 00 00 00 3E 00 00 00|1F 00 00 00 32 00 00 00 | >2
```

- Po odczytaniu jednej liczby: `fread(&x, sizeof(int), 1, plik);` wskaźnik jest automatycznie przesuwany o `sizeof(int)` bajtów

```
↓
25 00 00 00 1F 00 00 00|53 00 00 00 1B 00 00 00 | %S
06 00 00 00 3E 00 00 00|1F 00 00 00 32 00 00 00 | >2
```

- Po odczytaniu kolejnej liczby: `fread(&x, sizeof(int), 1, plik);` wskaźnik jest ponownie przesuwany o `sizeof(int)` bajtów

```
↓
25 00 00 00 1F 00 00 00|53 00 00 00 1B 00 00 00 | %S
06 00 00 00 3E 00 00 00|1F 00 00 00 32 00 00 00 | >2
```

- Plik binarny zawiera liczby: 37 31 83 27 6 62 31 50

## Rekordowe (blokowe) operacje wejścia-wyjścia

**REWIND** stdio.h  
`void rewind(FILE *fp);`

- Ustawia wskaźnik pozycji w pliku wskazywanym przez `fp` na początek pliku

**FTELL** stdio.h  
`long int ftell(FILE *fp);`

- Zwraca bieżące położenie w pliku wskazywanym przez `fp` (liczbę bajtów od początku pliku)

## Przykład: ile razy występuje w pliku wartość max

```
#include <stdio.h>

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

    fp = fopen("dane.dat", "rb");
    fread(&x, sizeof(int), 1, fp);
    max = x;
    while (!feof(fp))
    {
        if (x > max) max = x;
        fread(&x, sizeof(int), 1, fp);
    }
    printf("Wartosc max: %d\n", max);
}
```

```
7 3 3 0 3 9 6 4 1 8
6 0 4 5 4 9 4 5 4 5
9 9 8 0 0 5 3 5 1 0
```

## Przykład: ile razy występuje w pliku wartość max

```
rewind(fp);
fread(&x, sizeof(int), 1, fp);
while (!feof(fp))
{
    if (x == max) ile++;
    fread(&x, sizeof(int), 1, fp);
}
printf("Wystapienia max: %d\n", ile);
fclose(fp);
return 0;
}
```

```
7 3 3 0 3 9 6 4 1 8
6 0 4 5 4 9 4 5 4 5
9 9 8 0 0 5 3 5 1 0
```

```
Wartosc max: 9
Wystapienia max: 4
```

## Rekordowe (blokowe) operacje wejścia-wyjścia

**FSEEK** stdio.h  
`int fseek(FILE *fp, long int offset, int mode);`

- Pozwala przejść bezpośrednio do dowolnego bajtu w pliku wskazywanym przez `fp`
- `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`



## Przykład: odczytanie liczby o podanym numerze

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int x, nr;

    fp = fopen("dane.dat", "rb");
    printf("Nr: "); scanf("%d", &nr);
    while (fseek(fp, (nr-1)*sizeof(int), SEEK_SET) == 0)
    {
        fread(&x, sizeof(int), 1, fp);
        printf("Liczba: %d\n", x);
        printf("Nr: "); scanf("%d", &nr);
    }
    printf("Koniec!\n");
    fclose(fp);
    return 0;
}
```

```
7 3 3 0 3 9 6 4 1 8
6 0 4 5 4 9 4 5 4 5
9 9 8 0 0 5 3 5 1 0
```

```
Nr: 6
Liczba: 9
Nr: 14
Liczba: 5
Nr: 29
Liczba: 1
Nr: -1
Koniec!
```

## Rekordowe (blokowe) operacje wejścia-wyjścia

**FGETPOS** stdio.h  
`int fgetpos(FILE *fp, fpos_t *pos);`

- Zapamiętuje pod zmienną `pos` bieżące położenie w pliku wskazywanym przez `fp`; zwraca `0`, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd

**FSETPOS** stdio.h  
`int fsetpos(FILE *fp, const fpos_t *pos);`

- Przechodzi do położenia `pos` w pliku wskazywanym przez `fp`; zwraca `0`, gdy wywołania jest poprawne i wartość niezerową, gdy wystąpił błąd

Koniec wykładu nr 5

Dziękuję za uwagę!