

# Informatyka 2 (ES1D300 017)

Politechnika Białostocka - Wydział Elektryczny  
Elektrotechnika, semestr III, studia stacjonarne I stopnia  
Rok akademicki 2019/2020

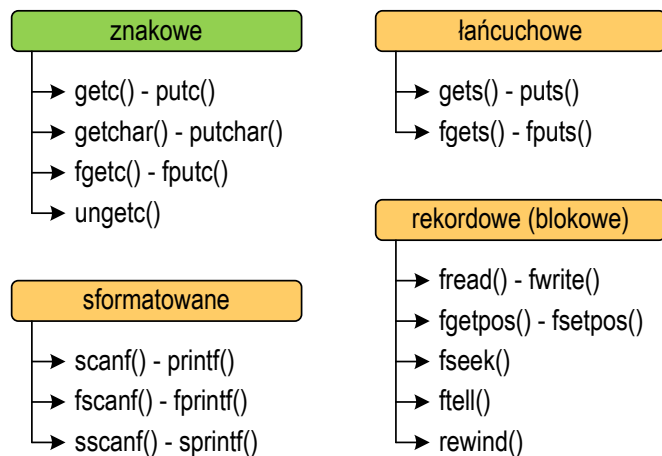
## Wykład nr 8 (19.11.2019)

dr inż. Jarosław Forenc

## Plan wykładu nr 8

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

## Znakowe operacje wejścia-wyjścia



## Znakowe operacje wejścia-wyjścia

```
GETCHAR stdio.h  
int getchar(void);
```

- Pobiera znak ze strumienia **stdin** (klawiatura)
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca przeczytany znak (typ **int**)
- Jeśli wystąpił błąd albo został przeczytany znacznik końca pliku, to funkcja zwraca wartość **EOF**

```
int znak;  
  
znak = getchar();  
printf("%c", znak);
```

## Znakowe operacje wejścia-wyjścia

```
PUTCHAR stdio.h  
int putchar(int znak);
```

- Wpisuje **znak** do strumienia **stdout** (standardowo ekran)
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca wypisany **znak**
- Jeśli wystąpił błąd, to funkcja zwraca wartość **EOF**

```
for (int i='a'; i<='z'; i++)  
    putchar(i);
```

```
abcdefghijklmnopqrstuvwxy
```

## Przykład: liczba znaków wczytanych z klawiatury

```
#include <stdio.h>  
  
int main(void)  
{  
    int znak, ile = 0;  
  
    while ((znak=getchar())!='\n')  
        ile++;  
  
    printf("Liczba znakow: %d\n",ile);  
  
    return 0;  
}
```

```
Ala ma laptopa  
Liczba znakow: 14
```

- Wprowadzane znaki są buforowane do naciśnięcia klawisza **Enter**
- Po naciśnięciu klawisza **Enter** zawartość bufora jest przesyłana do programu i analizowana w nim

## Znakowe operacje wejścia-wyjścia

```
FGETC stdio.h  
int fgetc(FILE *fp);
```

- Pobiera jeden znak ze strumienia wskazywanego przez **fp**
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca przeczytany znak po przekształceniu go na typ **int**
- Jeśli wystąpił błąd lub został przeczytany znacznik końca pliku, to funkcja zwraca wartość **EOF**

## Znakowe operacje wejścia-wyjścia

```
FPUTC stdio.h  
int fputc(int znak, FILE *fp);
```

- Wpisuje **znak** do otwartego strumienia reprezentowanego przez argument **fp**
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca wypisany **znak** (typ **int**)
- Jeśli wystąpił błąd, to funkcja zwraca wartość **EOF**

## Przykład: liczba wyrazów w pliku

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int znak, odstep = 1, ile = 0;

    fp = fopen("test.txt", "r");
    while ((znak = fgetc(fp)) != EOF)
        if (znak == ' ' || znak == '\t' || znak == '\n')
            odstep = 1;
        else
            if (odstep != 0) { odstep = 0; ile++; }
    fclose(fp);
    printf("Liczba slow: %d\n", ile);

    return 0;
}
```

Ala ma laptopa i psa.

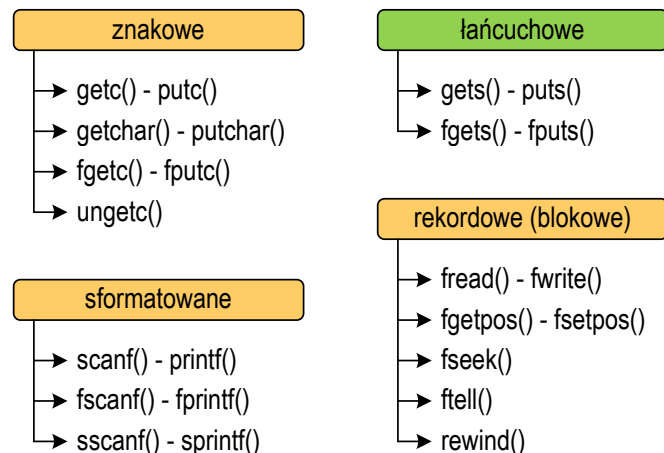
Liczba slow: 5

## Znakowe operacje wejścia-wyjścia

**UNGETC** stdio.h  
`int ungetc(int znak, FILE *fp);`

- Umieszcza **znak** z powrotem w strumieniu wejściowym **fp**

## 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()** umożliwia 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 CCRLF  
był język B,CRLF  
któryCRLF  
Ritchie rozwinał w język C.CRLF
```

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

```
Poprzednikiem języka CCRLF  
był język B,CRLF  
któryCRLF  
Ritchie rozwinał w język C.CRLF
```

## Przykład: wyświetlenie pliku tekstowego

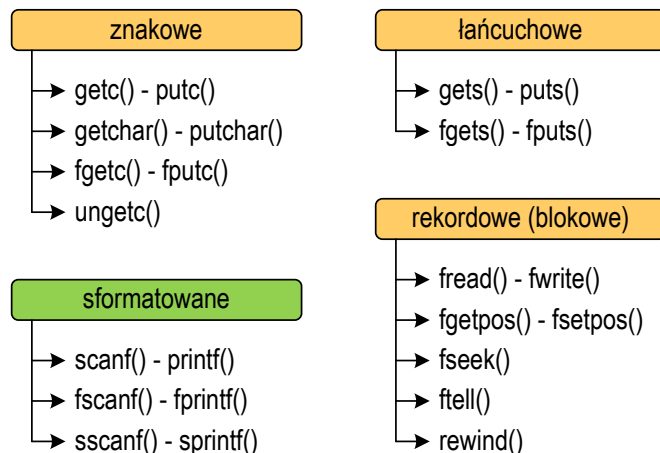
- Kolejne wywołania funkcji `fgets(buf,15,fp);` i zawartość tablicy `buf`

```
Poprzednikiem języka CCRLF  
był język B,CRLF  
któryCRLF  
Ritchie rozwinał w język C.CRLF
```

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

<sup>LF</sup> = `\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: 19
Wojciech Kowalski wiek: 22
Anna Jankowska wiek: 24
Krzysztof Mazur wiek: 29
Monika Krawczyk wiek: 24
Maja Piotrowska wiek: 21
Piotr Dudek wiek: 23
Julia Pawlak wiek: 22
```

## 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, 2019-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>
```

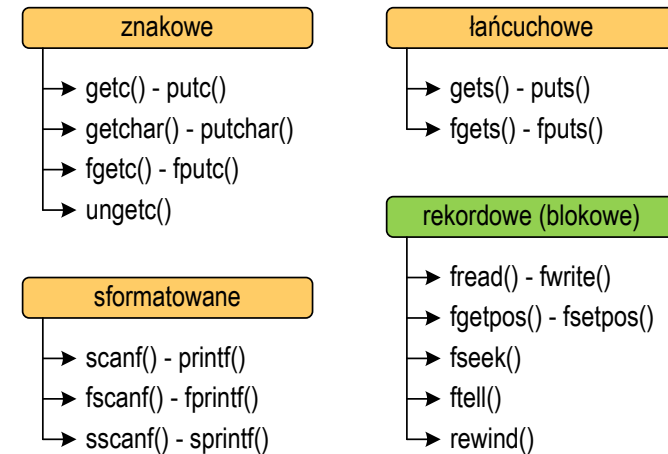
```
int main()
```

```
{  
    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, 2019-r);  
        fscanf(fp, "%s %s %d-%d-%d", naz, im, &d, &m, &r);  
    }  
    fclose(fp);  
    return 0;  
}
```

Grzegorz	Nowak	wiek: 19
Wojciech	Kowalski	wiek: 22
Anna	Jankowska	wiek: 24
Krzysztof	Mazur	wiek: 29
Monika	Krawczyk	wiek: 24
Maja	Piotrowska	wiek: 21
Piotr	Dudek	wiek: 23
Julia	Pawlak	wiek: 22

## Rekordowe (blokowe) operacje wejścia-wyjścia



## 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;  
}
```





## 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 określone 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 8

Dziękuję za uwagę!