



Fundusze Europejskie  
Wiedza Edukacja Rozwój



Rzeczpospolita  
Polska

Unia Europejska  
Europejski Fundusz Społeczny



Wydział Elektryczny  
Katedra Elektrotechniki Teoretycznej i Metrologii

Materiały do wykładu z przedmiotu:

**Informatyka**

**Kod: EDS1B1007**

**WYKŁAD NR 10**

**Opracował: dr inż. Jarosław Forenc**

**Białystok 2019**

Materiały zostały opracowane w ramach projektu „PB2020 - Zintegrowany Program Rozwoju Politechniki Białostockiej” realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

## Plan wykładu nr 10

- Format (plik) tekstowy i binarny
- Tryby otwarcia pliku: tekstowy i binarny
- Typy operacji wejścia-wyjścia
  - znakowe
  - łańcuchowe
  - sformatowane

## Operacje na plikach

- Operacje związane z przetwarzaniem pliku zazwyczaj składają się z trzech części

### 1. Otwarcie pliku (strumienia):

- funkcje: `fopen()`

### 2. Operacje na pliku (strumieniu), np. czytanie, pisanie:

- funkcje dla plików tekstowych: `fprintf()`, `fscanf()`, `fgetc()`,  
`fputc()`, `fgets()`, `fputs()`...

- funkcje dla plików binarnych: `fread()`, `fwrite()`, ...

### 3. Zamknięcie pliku (strumienia):

- funkcja: `fclose()`

## Operacje na plikach

```
#include <stdio.h>

int main(void)
{
    FILE *fp;

    fp = fopen("plik.txt", "w");
    if (fp == NULL)
    {
        printf("Bład otwarcia pliku.\n");
        return (-1);
    }

    /* przetwarzanie pliku */

    fclose(fp);

    return 0;
}
```

## Format (plik) tekstowy i binarny

- Przykład zawartości pliku tekstowego (**Notatnik**):

```
Plik (ang. file) - uporządkowany zbiór danych o skończonej długości,
posiadający szereg atrybutów i stanowiący dla użytkownika systemu
operacyjnego całość. Nazwa pliku nie jest częścią tego pliku,
lecz jest przechowywana w systemie plików.
```

- Przykład zawartości pliku binarnego (**Notatnik**):

```
MZ.....@.....č  .5. .Í!..
LÍ!This program cannot be run in DOS mode....$ {9ó?XF! ?XF! ?XF! !.ó! <X
f! !.Í! ,XF! !ž. ! =XF! ?Xg! \XF! !.â! 7XF! !.ň! >XF! !.÷! >XF! Rich?XF!
PE L. . ^ZR      í  7.σ.. 8 :      ↑◀. + + @ + 7 |
|          °.  J      L @. + + + + +      € . < .. $ |
. t.          . □L  W. . .textbss . +      í .text
```

## Format (plik) tekstowy i binarny

- Dane w pliku tekstowym zapisane są w postaci kodów ASCII
- Deklaracja i inicjalizacja zmiennej `x` typu `int`:

```
int x = 123456;
```

- W pamięci komputera zmienna `x` zajmuje 4 bajty:

00000000 00000001 11100010 01000000 (2)

- Po zapisaniu wartości zmiennej `x` do pliku **tekstowego** znajdzie się w nim 6 bajtów zawierających kody ASCII kolejnych cyfr

00110001 00110010 00110011 00110100 00110101 00110110 (2)

'1' '2' '3' '4' '5' '6' znaki

## Format (plik) tekstowy i binarny

- Dane w pliku tekstowym zapisane są w postaci kodów ASCII
- Deklaracja i inicjalizacja zmiennej `x` typu `int`:

```
int x = 123456;
```

- W pamięci komputera zmienna `x` zajmuje 4 bajty:

00000000	00000001	11100010	01000000
----------	----------	----------	----------

 (2)

- Po zapisaniu wartości zmiennej `x` do pliku `binarnego` znajdują się w nim 4 bajty o takiej samej zawartości jak w pamięci komputera

00000000	00000001	11100010	01000000
----------	----------	----------	----------

 (2)

## Format (plik) tekstowy i binarny

- Elementami pliku tekstowego są **wiersze** o różnej długości
- W systemach DOS/Windows każdy wiersz pliku tekstowego zakończony jest parą znaków:
  - **CR** (carriage return) - powrót karetki, kod ASCII -  $13_{(10)} = 0D_{(16)} = '\r'$
  - **LF** (line feed) - przesunięcie o wiersz, kod ASCII -  $10_{(10)} = 0A_{(16)} = '\n'$
- Załóżmy, że plik tekstowy ma postać:

```
Pierwszy wiersz pliku
Drugi wiersz pliku
Trzeci wiersz pliku
```

- Rzeczywista zawartość pliku jest następująca:

```
50 69 65 72 77 73 7A 79|20 77 69 65 72 73 7A 20 | Pierwszy wiersz
70 6C 69 6B 75 0D 0A 44|72 75 67 69 20 77 69 65 | pliku■■Drugi wie
72 73 7A 20 70 6C 69 6B|75 0D 0A 54 72 7A 65 63 | rsz pliku■■Trzec
69 20 77 69 65 72 73 7A|20 70 6C 69 6B 75 0D 0A | i wiersz pliku■■
```



## Format (plik) tekstowy i binarny

- W systemie Linux każdy wiersz pliku tekstowego zakończony jest tylko jednym znakiem:
  - **LF** (line feed) - przesunięcie o wiersz, kod ASCII -  $10_{(10)} = 0A_{(16)} = '\n'$
- Załóżmy, że plik tekstowy ma postać:

```
Pierwszy wiersz pliku
Drugi wiersz pliku
Trzeci wiersz pliku
```

- Rzeczywista zawartość pliku jest następująca:

```
50 69 65 72 77 73 7A 79|20 77 69 65 72 73 7A 20 | Pierwszy wiersz
70 6C 69 6B 75 0A 44 72|75 67 69 20 77 69 65 72 | plikuDrugi wier
73 7A 20 70 6C 69 6B 75|0A 54 72 7A 65 63 69 20 | sz plikuTrzeci
77 69 65 72 73 7A 20 70|6C 69 6B 75 0A | wiersz pliku
```

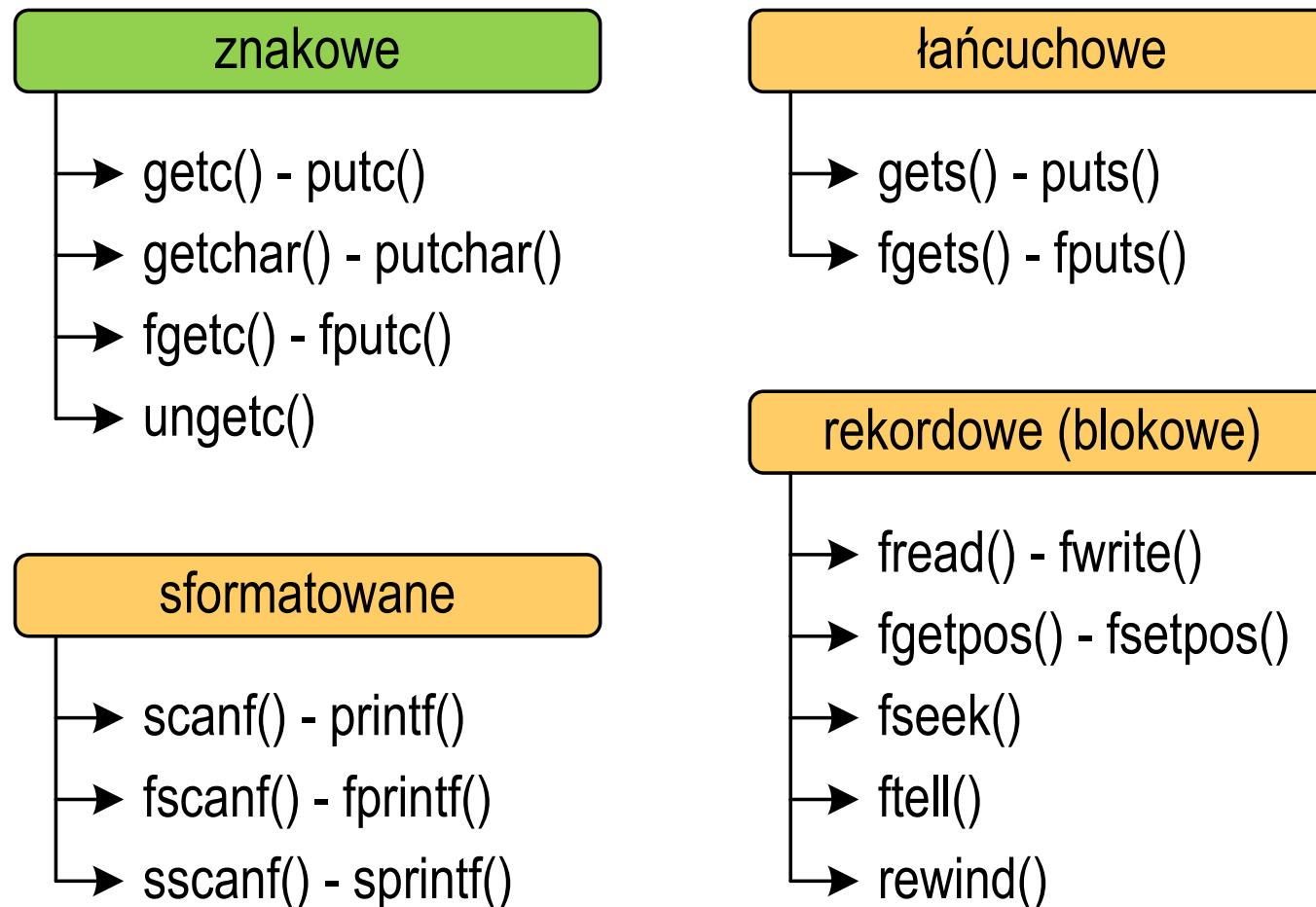
- Pliki **binarne** nie mają ściśle określonej struktury

## Tryby otwarcia pliku: tekstowy i binarny

```
FILE *fp1, *fp2;  
fp1 = fopen("dane.txt", "r"); // lub "rt"  
fp2 = fopen("dane.dat", "rb")
```

- Różnice pomiędzy trybem tekstowym i binarnym otwarcia pliku dotyczą innego traktowania znaków **CR** i **LF**
- W trybie **tekstowym**:
  - przy odczycie pliku para znaków **CR, LF** jest tłumaczona na znak nowej linii (**LF**)
  - przy zapisie pliku znak nowej linii (**LF**) jest zapisywany w postaci dwóch znaków (**CR, LF**)
- W trybie **binarnym**:
  - przy odczycie i zapisie para znaków **CR, LF** jest traktowana zawsze jako dwa znaki

## Znakowe operacje wejścia-wyjścia



## Znakowe operacje wejścia-wyjścia

GETC

stdio.h

```
int getc(FILE *fp);
```

- Pobiera jeden znak z aktualnej pozycji otwartego strumienia `fp` i uaktualnia pozycję
- Zmienna `fp` powinna wskazywać strukturę `FILE` reprezentującą strumień skojarzony z otwartym plikiem lub jeden ze standardowo otwartych strumieni (np. `stdin`)
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca wartość całkowitą `kodu` wczytanego znaku (typ `int`)
- Jeśli wystąpił błąd lub przeczytany został znacznik końca pliku, to funkcja zwraca wartość `EOF`

## Przykład: wyświetlenie pliku tekstowego

```
#include <stdio.h>

int main(void)
{
    FILE *fp;
    int znak;

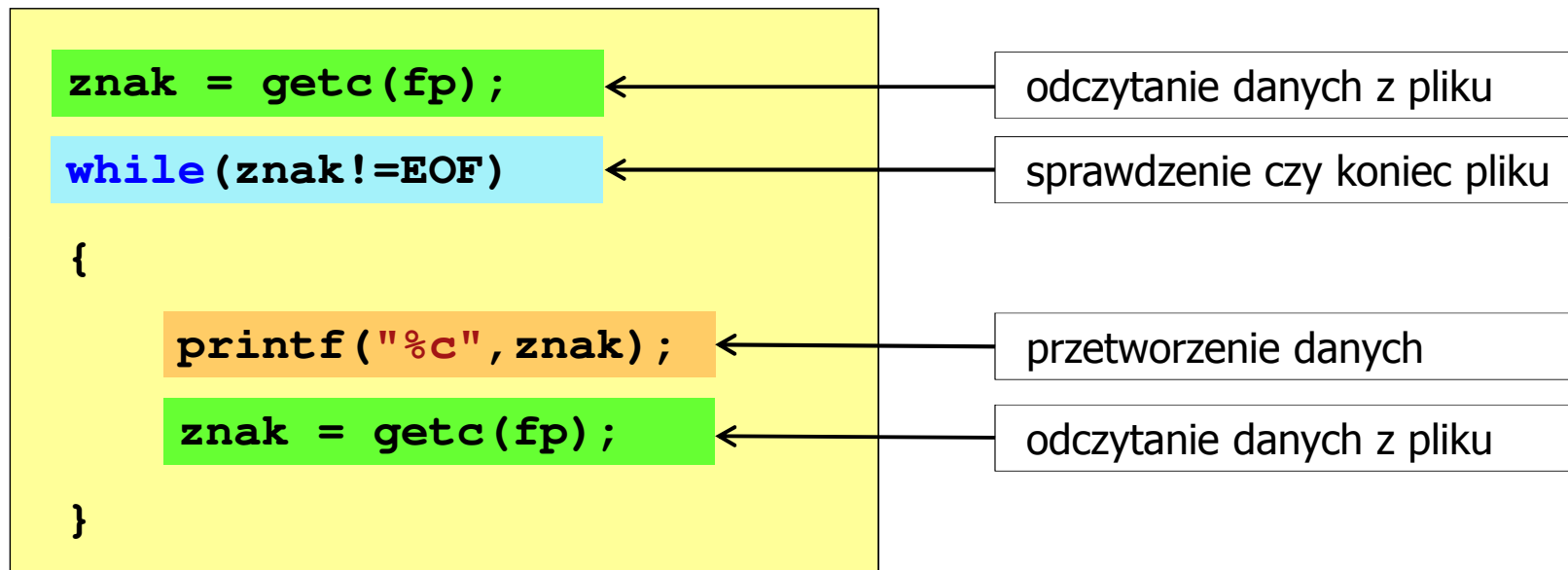
    fp = fopen("test.txt", "r");

    znak = getc(fp);
    while (znak != EOF)
    {
        printf("%c", znak);
        znak = getc(fp);
    }

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

## Schemat przetwarzania pliku

- Typowy schemat odczytywania danych z pliku



## Znakowe operacje wejścia-wyjścia

**putc**

**stdio.h**

```
int putc(int znak, FILE *fp);
```

- Wpisuje **znak** do otwartego strumienia reprezentowanego przez argument **fp**
- Zmienna **fp** powinna wskazywać strukturę **FILE** reprezentującą strumień skojarzony z otwartym plikiem lub jeden ze standardowo otwartych strumieni (np. **stdout**)
- Jeśli wykonanie zakończyło się poprawnie, to funkcja zwraca wypisany **znak**
- Jeśli wystąpił błąd, to funkcja zwraca wartość **EOF**

## Przykład: zapisanie alfabetu do pliku tekstowego

```
#include <stdio.h>

int main(void)
{
    FILE *fp = fopen("alfabet.txt", "w");

    for (int i='A'; i<='Z'; i++)
        putc(i, fp);

    fclose(fp);

    return 0;
}
```

ABCDEFGHIJKLMNOPQRSTUVWXYZ

- Stosując strumień `stdout` można wyświetlić alfabet na ekranie

```
for (int i='A'; i<='Z'; i++)
    putc(i, stdout);
```



## 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;

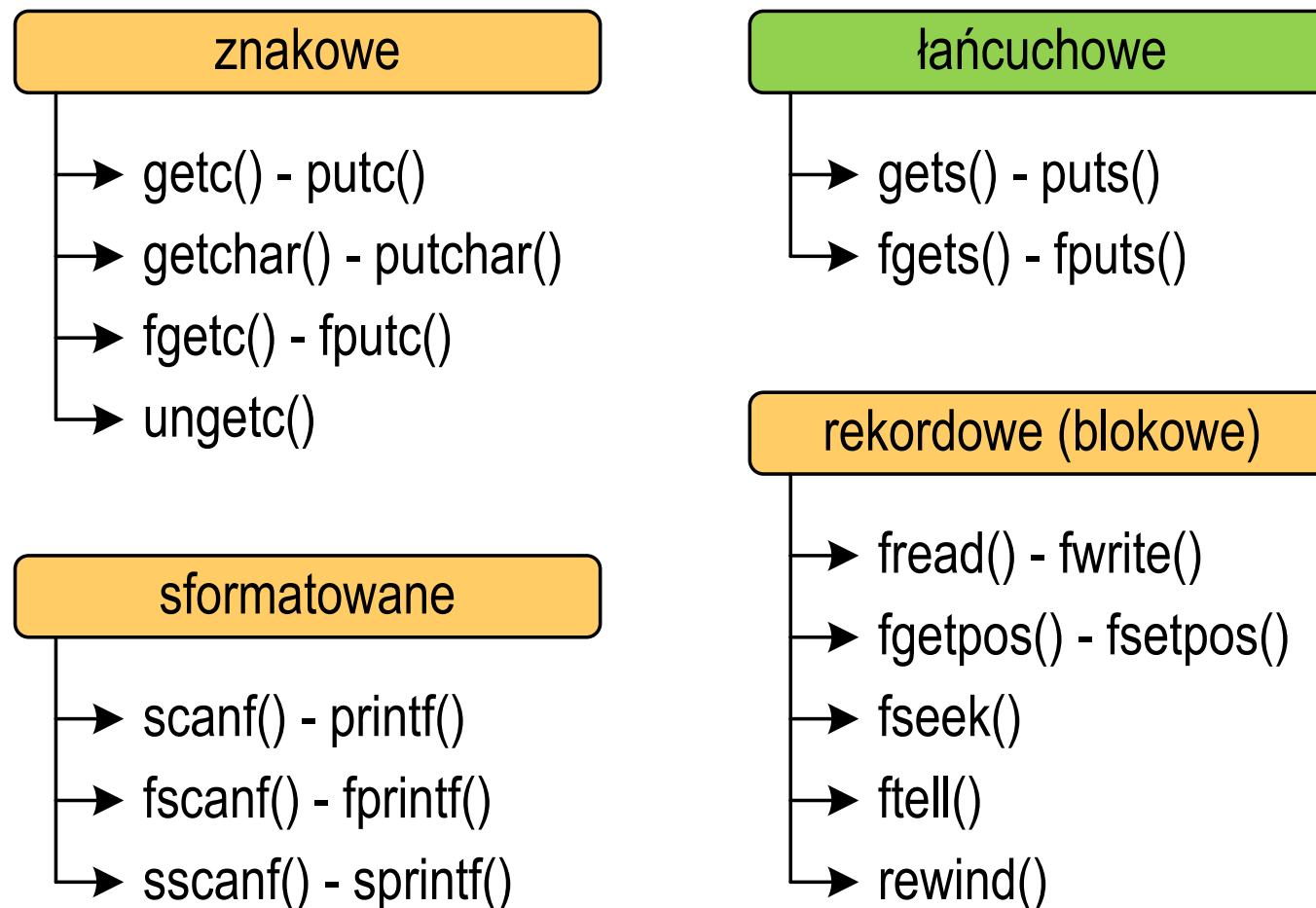
    fp = fopen("test.txt", "r");
    while ((znak = fgetc(fp)) != EOF)
        if (znak == ' ' || znak == '\t' || znak == '\n')
            odstep = 1;
        else
            if (odstep != 0) { odstep = 0; ile++; }
    fclose(fp);
    printf("Liczba slow: %d\n", ile);

    return 0;
}
```

Ala ma laptopa i psa.

Liczba slow: 5

## 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 C CR LF  
był język B, CR LF  
który CR LF  
Ritchie rozwinał w język C. CR LF
```

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

```
Poprzednikiem języka C CR LF  
był język B, CR LF  
który CR LF  
Ritchie rozwinał w język C. CR LF
```

## Przykład: wyświetlenie pliku tekstowego

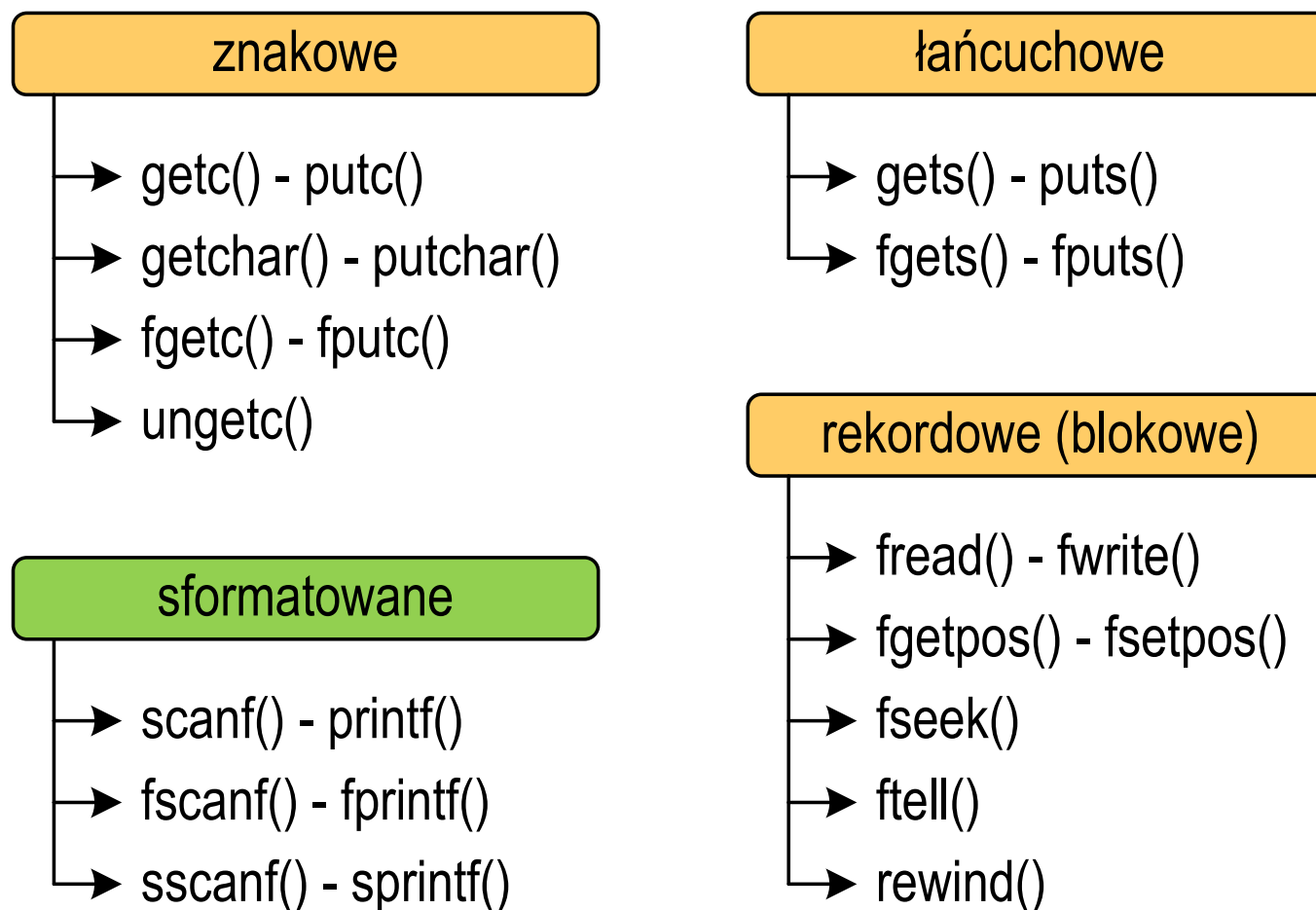
- Kolejne wywołania funkcji `fgets(buf,15,fp);` i zawartość tablicy `buf`

```
Poprzednikiem języka C CR LF  
był język B, CR LF  
który CR LF  
Ritchie rozwinał w język C. CR LF
```

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

`LF` = `\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]
```

Koniec wykładu nr 10

Dziękuję za uwagę!