

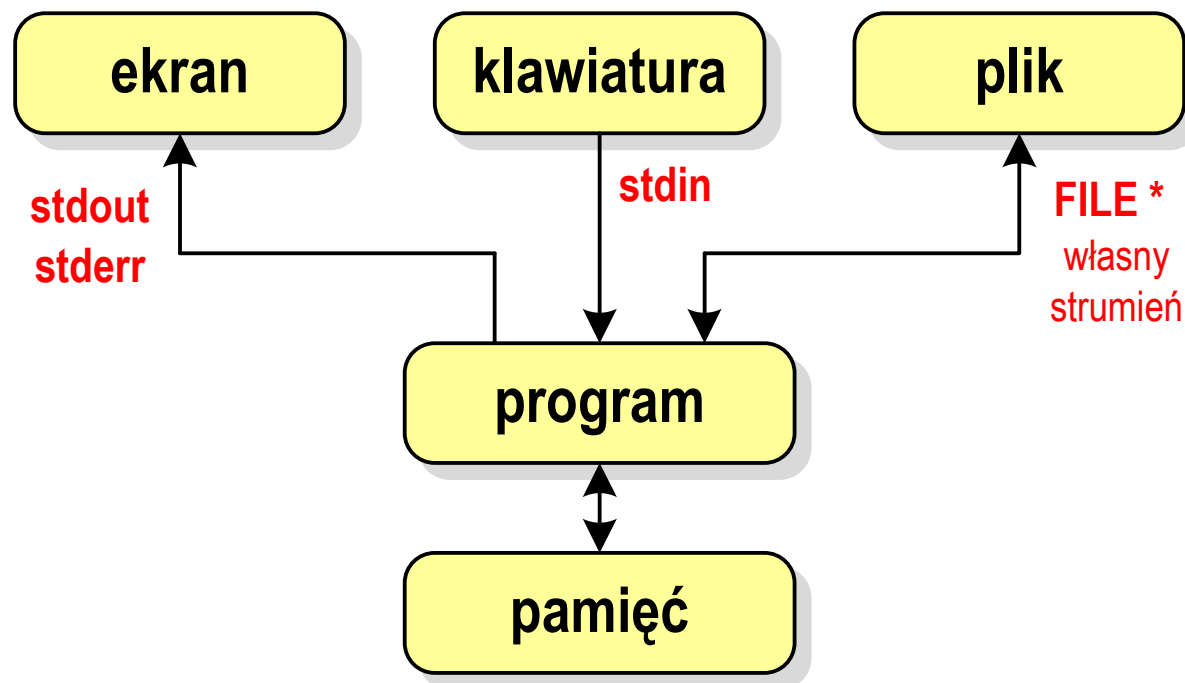
Programowanie w języku C (EAR1S02005)

Politechnika Białostocka - Wydział Elektryczny
Automatyka i Robotyka, semestr II, studia stacjonarne I stopnia
Rok akademicki 2020/2021

Zajęcia nr 10 (24.03.2021)

dr inż. Jarosław Forenc

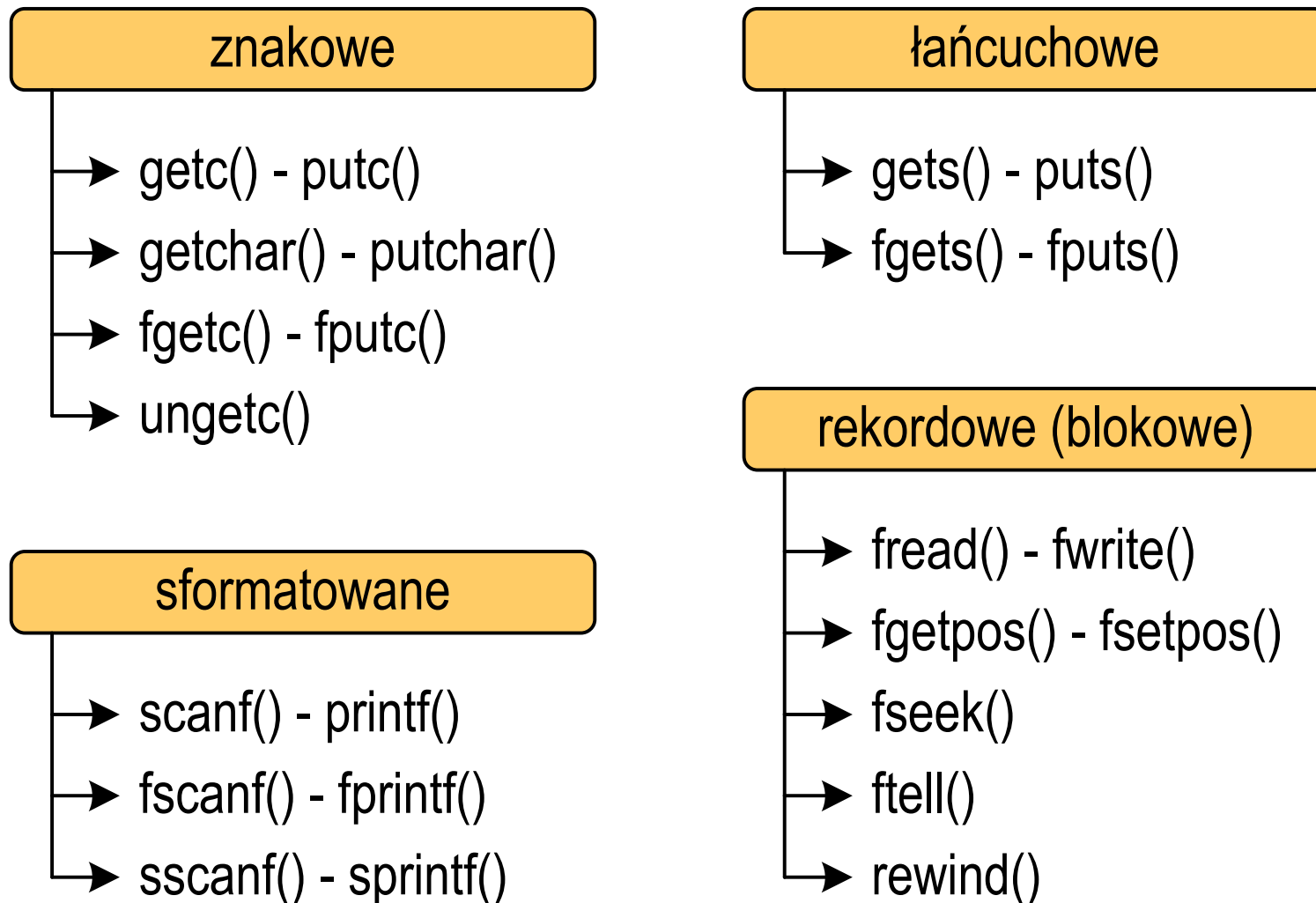
Operacje wejścia-wyjścia w języku C



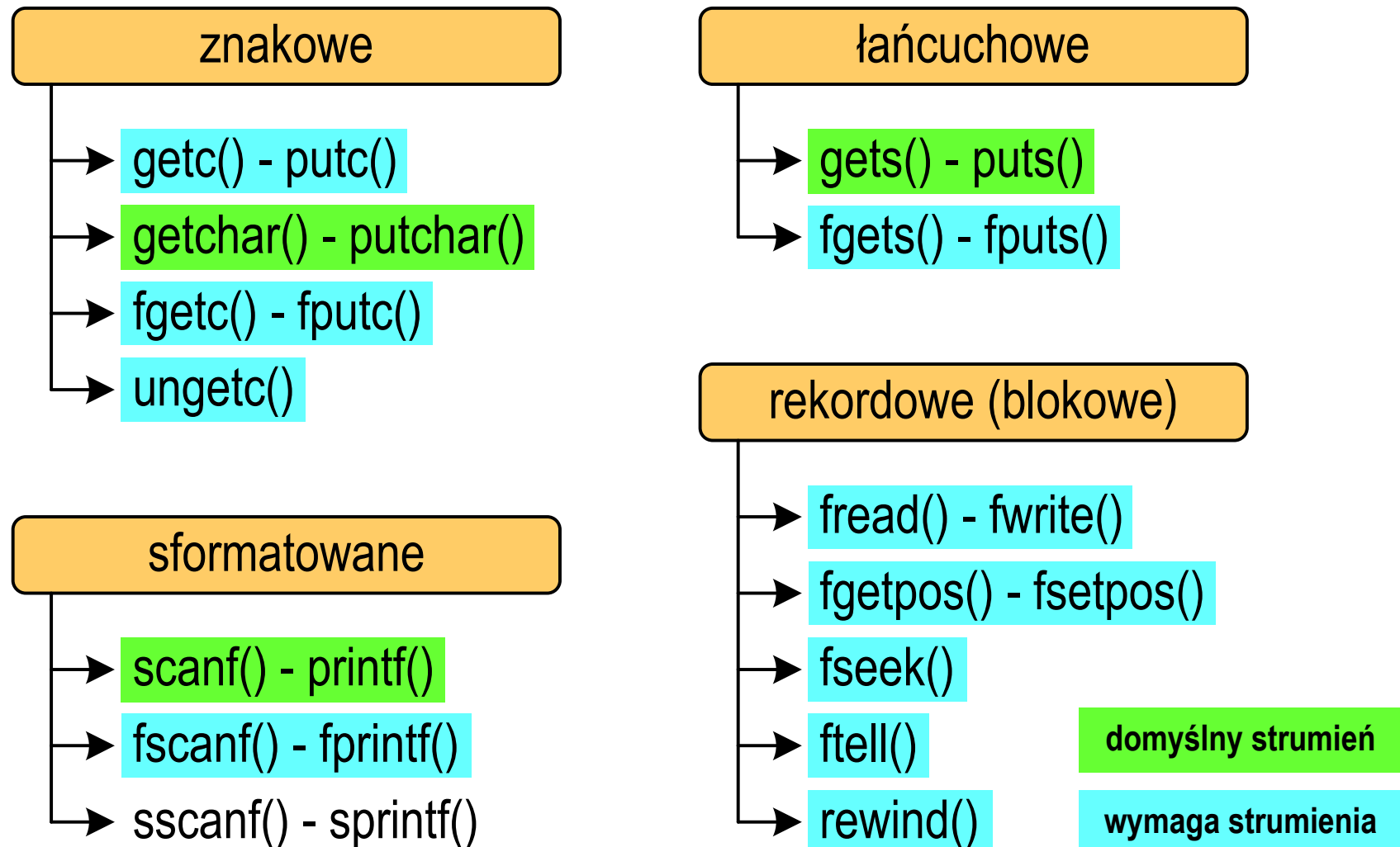
- **stdin** - standardowe wejście (klawiatura)
- **stdout** - standardowe wyjście (ekran)
- **stderr** - standardowe wyjście dla komunikatów o błędach (ekran)

- Standardowe funkcje wejścia-wyjścia mogą:
 - domyślnie korzystać z określonego strumienia (**stdin**, **stdout**, **stderr**)
np. **printf()** - **stdout**, **scanf()** - **stdin**
 - wymagać podania strumienia (własnego, **stdin**, **stdout**, **stderr**)

Typy standardowych operacji wejścia-wyjścia



Typy standardowych operacji wejścia-wyjścia



Znakowe operacje wejścia-wyjścia

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

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

- pobiera (czyta) jeden znak ze strumienia `fp` i zwraca jego kod (jako `int`) lub `EOF` (gdy napotkano koniec pliku)

```
int getchar();
```

- pobiera (czyta) jeden znak z klawiatury (strumienia `stdin`) i zwraca jego kod (jako `int`)

```
FILE *fp; int zn;  
zn = getc(fp);           // z pliku  
zn = fgetc(fp);         // z pliku  
zn = getchar();         // z klawiatury
```

Znakowe operacje wejścia-wyjścia

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

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

- wpisuje **znak** do otwartego strumienia **fp**

```
int putchar(int znak);
```

- wyświetla **znak** na ekranie (wpisuje do strumienia **stdin**)

```
FILE *fp; int zn = 'a';  
putc(zn, fp);           // do pliku  
fputc(zn, fp);         // do pliku  
putchar(zn);           // na ekran
```

Łańcuchowe operacje wejścia-wyjścia

```
char* gets(char *buf);
```

- czyta linię znaków z klawiatury (strumienia `stdin`) i zapisuje w tablicy `buf`; wczytywanie jest kończone po napotkaniu `'\n'`, który zastępowany jest znakiem `'\0'`

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

- czyta znaki z otwartego strumienia `fp` i zapisuje je w tablicy `buf`; przerywa pobieranie znaków po odczytaniu `'\n'` lub `max-1` znaków; zwraca `NULL` po napotkaniu końca pliku

```
FILE *fp; char txt[20];  
gets(txt);           // z klawiatury  
fgets(txt, 20, fp);  // z pliku
```

Łańcuchowe operacje wejścia-wyjścia

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

- wyświetla łańcuch znaków `buf` na ekranie (wpisuje do strumienia `stdout`);
zastępuje znak `\0` znakiem `\n`

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

- wpisuje znaki z tablicy `buf` do otwartego strumienia `fp`;
nie dołącza znaku końca wiersza `\n`

```
FILE *fp; char txt[20] = "Witaj swiecie";  
puts(txt);           // na ekran  
fputs(txt, fp);     // do pliku
```


Sformatowane operacje wejścia-wyjścia

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

- czyta dane z klawiatury (strumienia `stdin`)

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

- czyta dane z otwartego strumienia `fp` (najczęściej pliku)

```
int sscanf(FILE *fp, const char *format, ...);
```

- czyta dane z tablicy znaków `buf`

```
FILE *fp; char txt[30] = "15 3.14"; int x; float y;  
scanf("%d %f", &x, &y); // z klawiatury  
fscanf(fp, "%d %f", &x, &y); // z pliku  
sscanf(txt, "%d %f", &x, &y); // z tablicy znaków
```

Sformatowane operacje wejścia-wyjścia

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

- wyświetla dane na ekranie (wyprowadza do strumienia `stdout`)

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

- wyprowadza dane do otwartego strumienia `fp` (najczęściej pliku)

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

- zapisuje dane do tablicy znaków `buf`

```
FILE *fp; char txt[30];  
printf("Witaj świecie"); // na ekran  
fprintf(fp, "Witaj świecie"); // do pliku  
sprintf(txt, "Witaj świecie"); // do tablicy znaków
```

Operacje na plikach

- Strumień wiąże się z plikiem za pomocą **otwarcia**, zaś połączenie to jest przerywane przez **zamknięcie** strumienia
- 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()**

Otwarcie i zamknięcie pliku - `fopen()` / `fclose()`

```
FILE* fopen(const char *fname, const char *mode);
```

- `fname` - nazwa pliku, może zawierać całą ścieżkę dostępu do pliku
- `mode` - tryb otwarcia:
 - `"r"` - odczyt
 - `"w"` - zapis - jeśli pliku nie ma to zostanie on utworzony, jeśli plik istnieje, to jego poprzednia zawartość zostanie usunięta
 - `"a"` - zapis (dopisywanie) - dopisywanie danych na końcu istniejącego pliku, jeśli pliku nie ma to zostanie utworzony
 - `"t"` - otwarcie w trybie tekstowym (domyślnie)
 - `"b"` - otwarcie w trybie binarnym
- `fopen()` zwraca wskaźnik na strukturę `FILE` skojarzoną z otwartym plikiem lub `NULL`, gdy otwarcie nie powiodło się

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

- zamyka plik wskazywany przez `fp`

Przykład: otwarcie i zamknięcie pliku

```
#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;
}
```

Wykrycie końca pliku tekstowego

Funkcje	Metoda
getc(), fgetc()	zwracana wartość: EOF
fgets()	zwracana wartość: NULL
fscanf()	wywołanie funkcji: feof()

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

- zwraca wartość różną od zera, jeśli podczas ostatniej operacji odczytu pliku wskazywanego przez `fp` został wykryty jego koniec; w przeciwnym razie zwraca wartość `0` (zero)

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.

- dane w pliku tekstowym zapisane są w postaci kodów ASCII

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

```
MZ. L J .. @ c .s .Í! ,.
LÍ!This program cannot be run in DOS mode....$ {90ó?XF|?XF|?XF|! .ó|<X
f|! .Í! ,XF|! ž. |=XF|?Xg! LXF|! .â! 7XF|! .ň! >XF|! .÷! >XF|Rich?XF|
PE L. • .^ZR í 7.σ.. 8 : ↑◀. + + @ + 7 |
| . °. J L @. + + + + + € . < .. $|
. t. . □L W. . .textbss . + í .text
```

- dane w pliku binarnym zapisane są w takiej samej postaci jak w pamięci komputera

Operacje na plikach binarnych

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

- zapisuje **n** elementów o rozmiarze **s** bajtów każdy, do pliku określanego przez **fp**, biorąc dane z obszaru pamięci wskazywanego przez **p**
- zwraca liczbę faktycznie zapisanych elementów

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

- pobiera **n** elementów o rozmiarze **s** bajtów każdy, z pliku określanego przez **fp** i umieszcza odczytane dane w obszarze pamięci wskazywanym przez **p**
- zwraca liczbę faktycznie odczytanych elementów

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;
}
```

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:

