

# Procedūrinio programavimo pagrindai

Srautai

lekt. Irmantas Radavičius

[irmantas.radavicius@mif.vu.lt](mailto:irmantas.radavicius@mif.vu.lt)

Informatikos institutas, MIF, VU

# Turinys

---

Vartotojo sąsaja

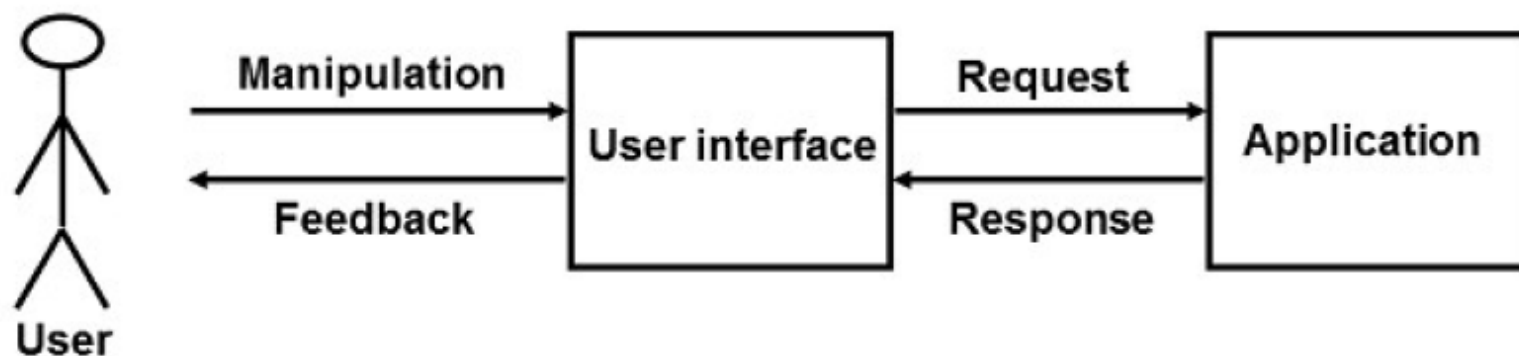
Srautai

I/O formatavimas

Duomenų validacija

# Vartotojo sąsaja

---



## Rekomendacijos

- aišku, ką programa daro
- aišku, kaip pateikti duomenis
- aišku, kur rasti rezultatus
- yra grįžtamasis ryšys
- programa bendrauja inteligentiškai

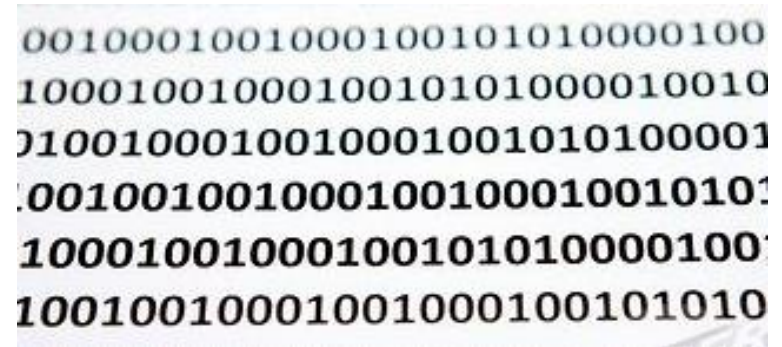
# Srautas

---

**Srautas** (angl. stream) – tai duomenų seka, atsirandanti laiko bėgyje.

**Dvejetainis** srautas – baitų seka.

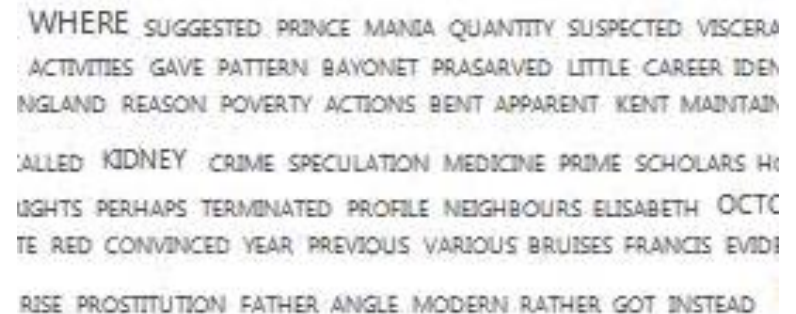
Baitai gali koduoti duomenis.



```
0010001001000100101010000100
1000100100010010101000010010
0100100010010001001010100001
0010010010001001000100101010
10001001000100101010000100
10010010001001000100101010
```

**Tekstinis** srautas – eilučių seka.

Eilutė – simbolių (baitų) seka,  
pasibaigianti '\n' .



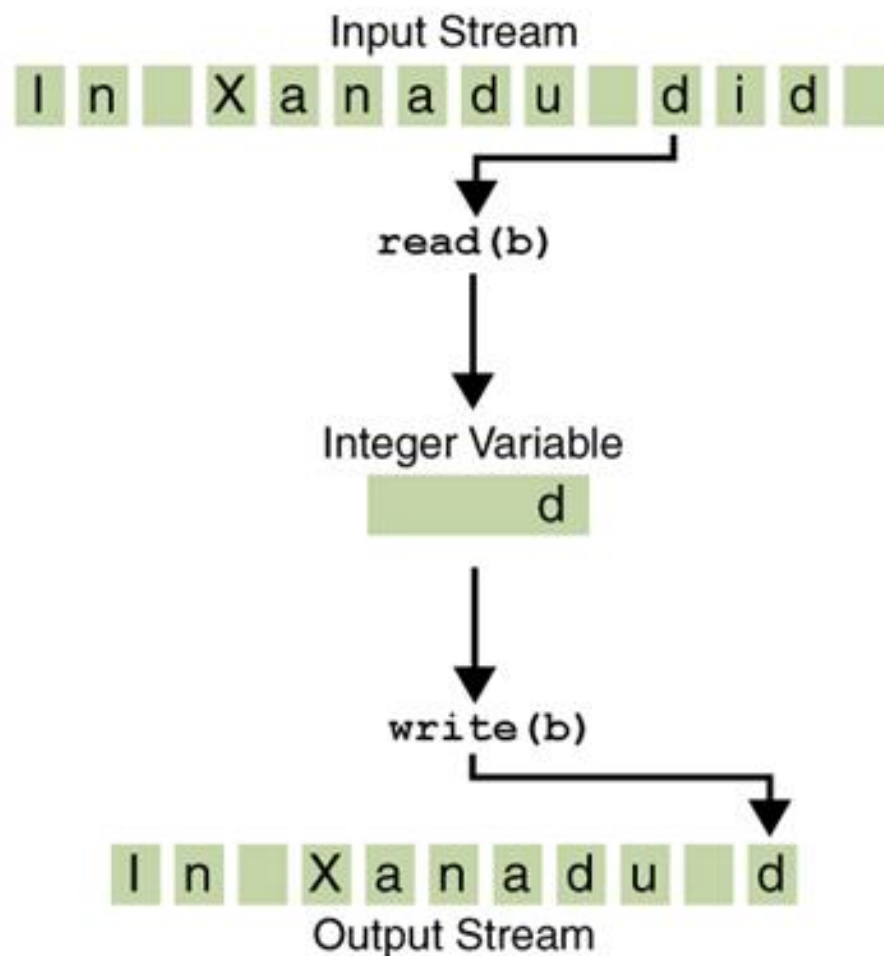
```
WHERE SUGGESTED PRINCE MANIA QUANTITY SUSPECTED VISCERA
ACTIVITIES GAVE PATTERN BAYONET PRASARVED LITTLE CAREER IDEN
INGLAND REASON POVERTY ACTIONS BENT APPARENT KENT MAINTAIN
ALLED KIDNEY CRIME SPECULATION MEDICINE PRIME SCHOLARS H
LIGHTS PERHAPS TERMINATED PROFILE NEIGHBOURS ELISABETH OCTO
TE RED CONVINCED YEAR PREVIOUS VARIOUS BRUISES FRANCIS EVIDE
RISE PROSTITUTION FATHER ANGLE MODERN RATHER GOT INSTEAD
```

# Duomenų tėkmė

## Duomenų tėkmės

(angl. data flow) kryptis:

- **išvedimo** sraute  
(angl. output stream) –  
iš programos
- **įvedimo** sraute  
(angl. input stream) –  
į programą



# I/O srautai

---

I/O srautai gali būti **susiejami** su:

- I/O įrenginiais (klaviatūra, ekranas, spausdintuvas, skaneris, etc.)
- failais

**Įrenginių failai** – OS priemonės,  
leidžiančios kreiptis į įrenginių tvarkykles (angl. driver)  
ir su jų pagalba kontroliuoti įrenginių darbą.

Ryšys sukuriamas **atidarant** srautą.

**Uždarant** srautą, ryšys nutraukiamas.

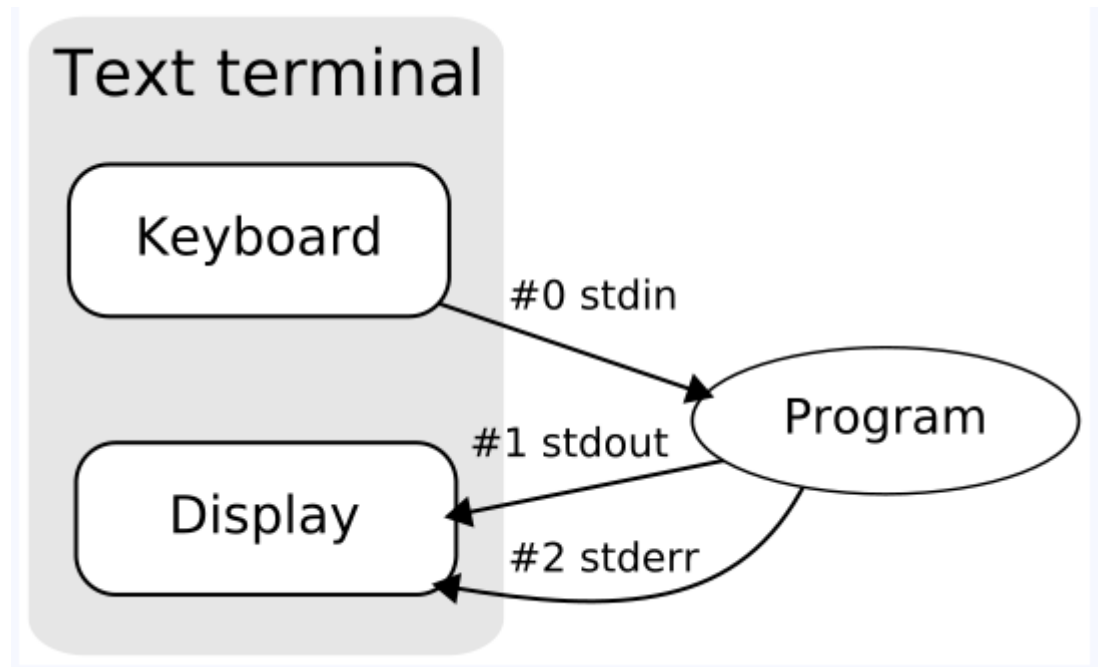
# C srautai

C kalboje visas I/O yra įgyvendinamas srautais

Iš anksto apibrėžti  
(angl. predefined)

**standartiniai** srautai:

- stdin
- stdout
- stderr



Pradėjus programą, jie jau būna atidaryti.

# Failo aprašymas

---

Darbui su srautais C kalboje naudojamos rodyklės į FILE tipo struktūrą.

**FILE struktūra** saugo informaciją,  
reikalingą failui (srautui) apdoroti.

Tai failo deskriptorius,  
informacija apie buferį (!),  
įvairūs požymiai, etc.

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

```
FILE *fp;
```



# Failo atidarymas

---

## ➤ Failo atidarymas

**FILE \*fopen(const char \*filename, const char \*mode)**

Vienu metu gali būti atidaryta ne daugiau nei FOPEN\_MAX failų.

Failo vardo ilgis negali viršyti FILENAME\_MAX.

Nesėkmės atveju grąžina NULL.

## Režimai

<b>"r"</b>	open text file for reading
<b>"w"</b>	create text file for writing; discard previous contents if any
<b>"a"</b>	append; open or create text file for writing at end of file
<b>"r+"</b>	open text file for update (i.e., reading and writing)
<b>"w+"</b>	create text file for update; discard previous contents if any
<b>"a+"</b>	append; open or create text file for update, writing at end

# Failo uždarymas

---

- Failo uždarymas!

```
int fclose(FILE *stream)
```

Sėkmės atveju grąžina 0, klaidos atveju – EOF.

- Failo pakartotinis atidarymas

```
FILE *freopen(const char *filename, const char *mode, FILE *stream)
```

Pirma bandoma uždaryti seną failą, tuomet bandoma atidaryti naują.

Naudojamas standartiniams srautams nukreipti.

# Failo požymiai

---

```
int feof(FILE *stream)
```

Jei pasiekta failo pabaiga, grąžina ne 0.

```
int ferror(FILE *stream)
```

Jei įvyko klaida, grąžina ne 0.

Panaudojimas – gavus EOF, atskirti klaidą nuo failo pabaigos.

```
void clearerr(FILE *stream)
```

Pašalinti failo pabaigos ir klaidų požymius FILE struktūroje.

```
void perror(const char *s)
```

```
fprintf(stderr, "%s: %s\n", s, "error message")
```

Atspausdinti s ir standartinį klaidos pranešimą (pagal kodą).

# Simbolių I/O

---

## ➤ Skaitymas iš failo

```
int fgetc(FILE *stream)
```

```
int getc(FILE *stream)
```

```
int getchar(void)
```

// EOF – klaida arba e.o.f.

// makrosas ??

// getc(stdin) ??

## ➤ Rašymas į failą

```
int fputc(int c, FILE *stream)
```

```
int putc(int c, FILE *stream)
```

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

// klaidos atveju EOF

// makrosas ??

// putc(stdout)

# Eilučių I/O

---

## ➤ Skaitymas

```
char *fgets(char *s, int n, FILE *stream)
```

Nustoja skaityti, jei (1) EOF (2) ‘\n’ (3) jau nuskaitytė n-1 simbolių

Pabaigoje (po ‘\n’) įrašo ‘\0’. Jei klaida ar pradžioje EOF, grąžina NULL.

```
char *gets(char *s)
```

Pakeičia ‘\n’ simboliu ‘\0’. Buferio dydis laikomas begaliniu!

## ➤ Rašymas

```
int fputs(const char *s, FILE *stream)
```

Klaidos atveju grąžina EOF, kitaip – neneigiamas skaičius.

```
int puts(const char *s)
```

Išveda s ir ‘\n’ į stdout.

# Formatuotas I/O

---

## ➤ Skaitymas

```
int fscanf(FILE *stream, const char *format, ...)
int scanf(const char *format, ...)           // fscanf(stdin, ...)
int sscanf(char *s, const char *format, ...) // skaitoma iš s
```

Funkcijos grąžina nuskaitytų argumentų skaičių.

## ➤ Rašymas

```
int fprintf(FILE *stream, const char *format, ...)
int printf(const char *format, ...)           // fprintf(stdout, ...)
int sprintf(char *s, const char *format, ...) // rašoma į s
```

Funkcijos grąžina įrašytų simbolių kiekį.

sprintf gale prideda '\0', bet jo neskaičiuoja.

# Printf funkcijų šeima

---

## Prototipai

```
int printf(char *format, arg1, arg2, ...)  
int fprintf(FILE *stream, const char *format, ...)  
int sprintf(char *s, const char *format, ...)
```

## Formato eilutės sudėtis

- paprastas tekstas
- formato specifikacijos

## Formato specifikacija

- prasideda %
- baigiasi konvertavimo specifikatoriumi (simboliu)

```
%<flags><field width><precision><length>conversion
```

# Sveikieji skaičiai

`%<flags><field width><precision><length>conversion`

Skaiciavimo sistemos:

<b>d, i</b>	int; decimal number.
<b>o</b>	int; unsigned octal number (without a leading zero).
<b>x, X</b>	int; unsigned hexadecimal number (without a leading 0x or 0X), using abcdef or ABCDEF for 10, ..., 15.
<b>u</b>	int; unsigned decimal number.

```
printf("%d %d %d\n", 64, 1000, -1 );  
printf("%i %i %i\n", 64, 1000, -1 );  
printf("%u %u %u\n", 64, 1000, -1 );  
printf("%x %x %x\n", 64, 1000, -1 );  
printf("%X %X %X\n", 64, 1000, -1 );  
printf("%o %o %o\n", 64, 1000, -1 );
```

```
64 1000 -1  
64 1000 -1  
64 1000 4294967295  
40 3e8 ffffffff  
40 3E8 FFFFFFFF  
100 1750 3777777777
```



# Dydis

%<flags><field width><precision><length>conversion

Nurodomas prieš formato specifikatorius d, i, u, x, X, o

h – short int arba unsigned short int

l – long int arba unsigned long int

```
printf("%hd %hd %hd %hd\n", 1, 65537, -1, 32768 );  
printf("%hu %hu %hu %hu\n", 1, 65537, -1, 32768 );
```

```
printf("%d %d %d %d\n", 1, 65537, -1, 32768 );  
printf("%u %u %u %u\n", 1, 65537, -1, 32768 );
```

```
printf("%ld %ld %ld %ld\n", 1, 65537, -1, 32768 );  
printf("%lu %lu %lu %lu\n", 1, 65537, -1, 32768 );
```

```
1 1 -1 -32768  
1 1 65535 32768  
  
1 65537 -1 32768  
1 65537 4294967295 32768  
  
1 65537 -1 32768  
1 65537 4294967295 32768
```

# Slankaus kablelio skaičiai

`%<flags><field width><precision><length>conversion`

Double (!) tipo kintamojo išvedimo formatai.

f – formatas `[-]mmm.ddd`

e, E – formatas `[-]m.dddddde±xx` `[-]m.ddddddE±xx`

g, G – formatas priklauso nuo laipsnio, f arba e, f arba E

Jei nurodyta su  
dydžio specifikatoriumi L,  
tuomet tipas –  
long double.

```
printf("%f %f %f\n", 0.1, 0.000001, 0.0000001 );  
printf("%e %e %e\n", 0.1, 0.000001, 0.0000001 );  
printf("%E %E %E\n", 0.1, 0.000001, 0.0000001 );  
printf("%g %g %g\n", 0.1, 0.000001, 0.0000001 );  
printf("%G %G %G\n", 0.1, 0.000001, 0.0000001 );
```

```
0.100000 0.000001 0.000000  
1.000000e-001 1.000000e-006 1.000000e-007  
1.000000E-001 1.000000E-006 1.000000E-007  
0.1 1e-006 1e-007  
0.1 1E-006 1E-007
```

# Plotis ir tikslumas

---

`%<flags><field width><precision><length>conversion`

Plotį nuo tikslumo skiria taškas.

Plotis nurodo minimalų (!) lauko dydį. Esant reikalui, įdedami tarpai.

Tikslumas nurodo, kiek simbolių spausdinti.

realiems skaičiams – skaitmenų skaičius po kablelio

sveikiems skaičiams – minimalus skaičiaus ilgis (užpildoma nuliais)

```
printf("%7.1f %7.2f %7.3f %7.4f\n", 0.1234, 0.1234, 0.1234, 0.1234 );  
printf("%7.1d %7.2d %7.3d %7.4d\n", 1, 2, 3, 4 );
```

```
    0.1    0.12    0.123    0.1234  
     1     02     003     0004  
1234567
```

# Plotis ir tikslumas

`%<flags><field width><precision><length>conversion`

Formatas *g*, *G* taiko formatą *e*, *E*, jei laipsnis mažesnis už -4 arba laipsnis nėra mažesnis už tikslumą; kitu atveju, taiko formatą *f*.

```
printf("%g %g\n", 0.0001, 0.00001);  
printf("%g %g\n", 111111.0, 111111.0);
```

```
0.0001 1e-005  
111111 1.11111e+006
```

Nulis po kablelio reikalingas!

Plotį ir tikslumą galima nurodyti printf parametrų sąrašė.

```
printf("%*.*d %*.*d\n", 4, 1, 0, 3, 2, 0);  
printf("%4.1d %3.2d\n", 0, 0);
```

```
0 00  
0 00  
1234 123
```

# Vėliavėlės

`%<flags><field width><precision><length>conversion`

Vėliavėlės gali būti nurodytos bet kuria tvarka.

- lygiuoti pagal kairįjį kraštą
- 0 vietoje tarpų kairėje pusėje užpildyti nuliais
- + bet kuriuo atveju rašyti ženklą
- space palikti vietą ženklui (jei jo nėra)

```
printf("%6d %+1d %06d\n", 123, 12, 1);  
printf("%-6d % 1d %6d\n", 123, 12, 1);
```

```
123 +12 000001  
123 12 1  
123456 123 123456
```

# Vēliavēlēs

%<flags><field width><precision><length>conversion

Vēliavēlē # nurodo taikyti alternatīvų išvedimo formatą.

#o #x #X      pradžia bus atitinkamai 0, 0x arba 0X

#f #e #E      būtinai išvesti tašką

#g #G      būtinai išvesti tašką ir nepašalinti nulių pabaigoje

```
printf("%6o %4x %4X\n", 1, 1, 1);
printf("%#6o %#4x %#4X\n", 1, 1, 1);

printf("%6.0f %8.0e %8.0E\n", 1.0, 1.0, 1.0);
printf("%#6.0f %#8.0e %#8.0E\n", 1.0, 1.0, 1.0);

printf("%6.2g %6.2G\n", 1.0, 1.0);
printf("%#6.2g %#6.2G\n", 1.0, 1.0);
```

```
1      1      1
01     0x1     0X1

1      1e+000   1E+000
1.     1.e+000  1.E+000

1      1
1.0    1.0
```

# Tekstas

`%<flags><field width><precision><length>conversion`

## Teksto išvedimas

<b>c</b>	<b>int</b> ; single character, after conversion to unsigned char.
<b>s</b>	<b>char *</b> ; characters from the string are printed until a <code>'\0'</code> is reached or until the number of characters indicated by the precision have been printed.

```
printf(":%s:\n", "Hello, world!");
printf(":%15s:\n", "Hello, world!");
printf(":%.10s:\n", "Hello, world!");
printf(":%-10s:\n", "Hello, world!");
printf(":%-15s:\n", "Hello, world!");
printf(":%.15s:\n", "Hello, world!");
printf(":%15.10s:\n", "Hello, world!");
printf(":%-15.10s:\n", "Hello, world!");
```

```
:Hello, world!:
:  Hello, world!:
:Hello, wor:
:Hello, world!:
:Hello, world! :
:Hello, world!:
:      Hello, wor:
:Hello, wor      :
```

# Kiti

---

`%<flags><field width><precision><length>conversion`

## Kiti specifikatoriai

<b>p</b>	<b>void *</b> ; print as a pointer (implementation-dependent representation).
<b>n</b>	<b>int *</b> ; the number of characters written so far by this call to <b>printf</b> is <i>written into</i> the argument. No argument is converted.

```
int a, *m = (int *)malloc(1);  
printf("%12p %12p %12p\n", &a, m, &m);  
free(m);
```

```
0022FF44  00280F30  0022FF40
```



# Scanf funkcijų šeima

---

## Prototipai

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

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

```
int sscanf(char *s, const char *format, ...)
```

## Formato eilutės sudėtis

- “whitespace”
- paprasti simboliai
- formato specifikacijos

# Taisyklės

---

**“Whitespace” simbolis** (tarpas, eilutės pabaiga, etc.) nurodo, jog įvedimo sraute bus praleidžiami “whitespace” simboliai iki kito “non-whitespace” simbolio.

**Paprastas simbolis** nurodo, jog būtent tokį simbolį bus tikimasi gauti įvedimo sraute. Jei toks randamas, skaitymas tęsiamas, jei ne – funkcija grąžina nuskaitytą (netikusį) simbolį į srautą ir sustoja.

**Formato specifikacijos** (prasidedančios %) nurodo, jog bus bandoma atpažinti ir tada arba nuskaityti, arba praleisti įvedimo sraute esančią tam tikro tipo (formato) reikšmę.

# Taisyklės

---

Formato specifikacijos struktūra:

`%[*][width][modifiers]type`

Formato specifikacija nusako, kaip bus traktuojama reikšmė sraute, kuri suprantama kaip “non-whitespace” simbolių seka (yra išimčių), atitinkanti formatą (tipą) ir ne ilgesnė už nurodytą maksimalų plotį.

Nuskaityta reikšmė yra priskiriama atitinkamo tipo parametrai.

Jei nurodytas požymis \*, priskyrimas nevyksta, reikšmė praleidžiama.

# Skaitiniai tipai

d, i, o, u, x –

nuskaitomos sveikojo tipo reikšmės

e, f, g –

nuskaitomos float tipo (!) reikšmės

Specifier	Modifies	Converts
l	d i o u x	long int
h	d i o u x	short int
l	e f	double
L	e f	long double

```
float f; double d; long double ld;
scanf("%f %f %f", &f, &d, &ld);
printf("%f %f %Lf\n", f, d, ld);
scanf("%lf %lf %lf", &f, &d, &ld);
printf("%f %f %Lf\n", f, d, ld);
scanf("%Lf %Lf %Lf", &f, &d, &ld);
printf("%f %f %Lf\n", f, d, ld);
```

```
Input: 1.0 1.0 1.0
1.000000 0.000000 0.000000
Input: 1.0 1.0 1.0
0.000000 1.000000 1.000000
Input: 1.0 1.0 1.0
0.000000 1.000000 1.000000
```

# Skaitiniai tipai

---

Kiti specifikatoriai (analogiškai printf):

p – nuskaitymas adresas

n – grąžina (įrašo) šiuo kreipiniu iki šiol nuskaitytų simbolių skaičių.

Maksimalus plotis

```
int i1 = -1, i2 = -1, i3 = -1;  
scanf("%2d %4d %6d", &i1, &i2, &i3);  
printf("%d %d %d\n", i1, i2, i3);
```

```
Input: 0123456789abc  
1 2345 6789
```

Reikšmių filtravimas

```
int i1 = -1, i2 = -1, i3 = -1;  
scanf("%2d %*4d %6d", &i1, &i2, &i3);  
printf("%d %d %d\n", i1, i2, i3);
```

```
Input: 0123456789abc  
1 6789 -1
```

# Tekstas

---

c –

nuskaitomas simbolis (išimtis: įskaitant ir “whitespace”)  
naudojant reikšmės dydį, galima nuskaityti simbolių masyvą

s –

nuskaitomas tekstas (žodis – “whitespace” skiria žodžius)  
gale pridedamas ‘\0’

```
char c1, c2[10] = "something", *c3 = (char *)malloc(10), c4[20];  
scanf("%c%5c%s%10s", &c1, c2, c3, c4);
```

```
Input: Hello world, hellohelloworld?  
Output:
```

```
c1  'H'  
c2  "ello hing"  
c3  "world,"  
c4  "hellohello"
```

# Simbolių aibė

---

Nuskaitant tekstą, galima išreikštinau nurodyti tinkamų ar netinkamų simbolių aibę (angl. scan set).

Tinkami simboliai [...]

Netinkami simboliai [^...]

Aibės viduje galima nurodyti režius [start]-[end]

Minusas turėtų būti pirmas arba paskutinis simbolis aibėje.

```
char str[128];  
scanf("%[A-Z]", str);  
scanf("%[^.,]", str);  
scanf("%127[^\\n]", str);
```

```
HELLO world  
Entered: HELLO  
Hello, world  
Entered: Hello  
Hello world, hello world  
Entered: Hello world, hello world
```

# Tekstiniai ir binariniai failai

---

**Tekstinių failų** elementas – eilutė su pabaigos ženklu ‘\n’.

Tekstiniai failai yra įskaitomi žmogui.

**Binarinių failų** elementas – fiksuoto dydžio įrašas.

Binariniai failai leidžia tiesioginę prieigą prie bet kurio įrašo.

**Skirtumai** tarp jų: eilučių pabaigos požymių traktavimas, etc.

Kai kuriose sistemose skirtumo nėra!

Binariniai failai sukuriama funkcijai fopen nurodžius režimą ‘b’.



# Pozicionavimas faile

---

```
long ftell(FILE *stream)
```

Grąžina einamąją poziciją faile.

```
int fseek(FILE *stream, long offset, int origin)
```

Nustato poziciją faile (per poslinkį) nuo atskaitos taško.

Atskaitos taškas: SEEK\_SET – pradžia,

SEEK\_CUR – einamoji pozicija, SEEK\_END – pabaiga.

Tekstiniam failams arba poslinkis turi būti arba 0,  
arba ftell rezultatas (tada naudotinas su SEEK\_SET).

```
void rewind(FILE *stream)
```

```
fseek(fp, 0L, SEEK_SET); clearerr(fp).
```

# Blokų I/O

---

## ➤ Skaitymas

**size\_t fread(void \*ptr, size\_t size, size\_t nobj, FILE \*stream)**

Nuskaito ne daugiau nei nobj blokų.

Grąžina nuskaitytų blokų skaičių.

Jei jis mažesnis nei norėta, naudojamos ferror, feof.

## ➤ Rašymas

**size\_t fwrite(const void \*ptr, size\_t size, size\_t nobj, FILE \*stream)**

Grąžina įrašytų blokų skaičių.

Jei jis mažesnis nei norėta, įvyko klaida.

# Duomenų validacija

---

Programa privalo:

- ✓ nelūžti
- ✓ visais atvejais duoti korektišką rezultatą

Duomenų validacija yra privaloma!

Tikrinama:

- ✓ duomenų tipas
- ✓ režiai
- ✓ formatas

Atsparumo nemokšoms (angl. foolproof) sąvoka

```
printf("%-10s\n", "Bye!");
```