

Algoritmizácia a programovanie

1. prednáška

Ján Grman



Harmonogram



- Prednáška - Pondelok 8:00
- Cvičenia
 - Pondelok 10:00, 15:00, 17:00
 - Utorok 08:00, 10:00, 13:00
 - Piatok 10:00, 13:00, 15:00

Obsah prednášky



Kontakt:

Mgr. Ján Grman, PhD.

Ústav elektrotechniky – B322

jan.grman@stuba.sk

1. Úvod do predmetu
2. Algoritmy a procedurálna paradigma
3. Základy programovania v jazyku C

1. Úvod



Cieľ predmetu



- získať základné znalosti z tvorby algoritmov v rámci procedurálnej a čiastočne objektovo orientovanej paradigmy
- naučiť sa základné konštrukcie jazyka C
- získať zručnosti v tvorbe vybraných algortimov a programov v jazyku C

Náplň predmetu



1. základné konštrukcie programovacieho jazyka
2. funkcie, iterácia a rekurzia
3. základné údajové štruktúry
4. práca so súbormi
5. práca s jednorozmernými poliami a reťazcami
6. preprocesor jazyka C

Kde hľadať informácie?



- prednášky
- cvičenia
- literatúra:
 - WIRTH, N. Algoritmy a štruktúry údajov. Bratislava: Alfa, 1989. 481 s. ISBN 80-05-00153-3.
 - KERNIGHAN, B W. – RITCHIE, D M. Programovací jazyk C. Brno: Computer Press, 2006. 286 s. ISBN 80-251-0897-X.
 - HEROUT, P. Učebnice jazyka C : 1. díl. České Budějovice: Nakladatelství KOPP, 2009. 271 s. ISBN 978-80-7232-383-8.
 - VIRIUS, M. Jazyky C a C++ : Kompletní průvodce. Praha: Grada Publishing, 2011. 367 s. ISBN 978-80-247-3917-5.
 - PROKOP, J. Algoritmy v jazyku C a C++ -2, rozšířené a aktualizované vydání. Praha: Grada Publishing sa.s., 2012.
 - T. Ward, G. Dodrill: C Language Tutorial, 1999,
URL: <http://phy.ntnu.edu.tw/~cchen/ctutor.pdf>
 - iné: literatúra, Internet (?)

Prednášky



- **Prečo je dôležité sledovať prednášky a konzultovať:**
 - veľa z povedaného nie je v prezentáciách
 - príklady, programy – aj mimo prezentácií
 - diskusia so študentami

Cvičenia



- Riešenie úloh
 - treba úlohy riešiť aktívne a samostatne
 - vedieť vyriešiť všetky úlohy = byť pripravený na skúšku
- **Aktívna účasť na cvičeniach a CELÝ SEMESTER !!!**
 - študent je povinný aktívne pracovať na zadaniach
 - plagiátorstvo bude aktívne kontrolované a prísne sankcionované
- **Využite konzultácie už v prípade prvých problémov!**
- Prostredie: **CodeBlocks**, povolené je ľubovoľné prostredie podporujúce ANSI C (používame výhradne ANSI C), ale kontrola zadaní bude v CodeBlocks.

2. Algoritmy a paradigmy programovania



Algoritmus



- predpis, metóda alebo technika, ktorá špecifikuje postup úkonov potrebných na dosiahnutie riešenia nejakej úlohy
 - napr. usporiadanie zoznamu mien podľa abecedy
 - napr. recept na bábovku
- *v informatike*: je jednoznačná, presná a konečná postupnosť operácií, ktoré sú aplikovateľné na množinu objektov alebo symbolov (čísiel, šachových figúrok, surovín na bábovku)
 - počiatočný stav týchto objektov je vstupom
 - ich koncový stav je výstupom
 - počet operácií, vstupy a výstupy sú konečné (aj keď počítame napr. s iracionálnym číslom π)

Počítačový program



- konkrétna reprezentácia algoritmu v nejakom programovacom jazyku

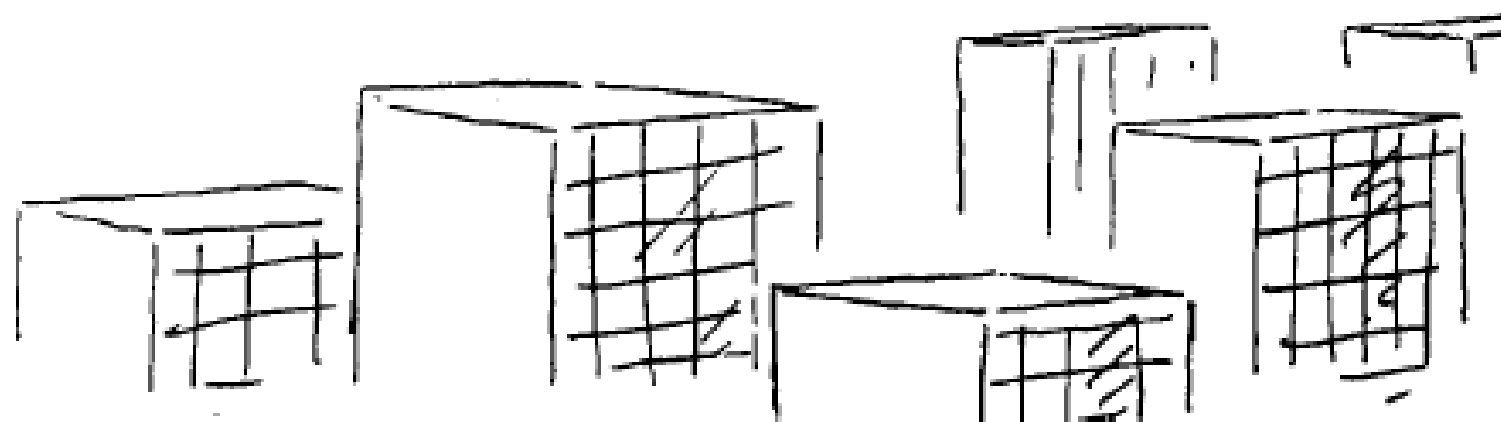
Procedurálne programovanie



- procedúry a riadiace štruktúry, napr. cykly, podmienky
- jazyky: C, PASCAL, Pyton, C#, Java
- program je postupnosť príkazov
- príkazy predpisujú vykonanie operácií
 - ak neurčí riadiaca štruktúra inak, vykonajú sa v tej postupnosti, v akej sú zapísané
- jazykové konštrukcie pre
 - vetvenie (napr. príkaz **if**, **case**, **switch**)
 - cyklus (napr. príkaz **for**, **while**, **repeat-until**, **do-while**).

3. Úvod do programovania





Ked' nepoznajú Céčko, sme stratení!

Jazyk C



- je univerzálny programovací jazyk nízkej úrovne
 - pracuje len so štandardnými dátovými typmi (znak, celé číslo, reálne číslo...)
- má úsporné vyjadrovanie
- pre mnohé úlohy je efektívnejší a rýchlejší ako iné jazyky
- bol navrhnutý a implementovaný pod operačným systémom UNIX

Jazyk C



- jazyk nízkej úrovne
 - priamo neumožňuje prácu s reťazcami
 - všetky akcie s reťazcami - pomocou funkcií (v knižniciach)
- výhody
 - jednoduchosť
 - nezávislosť na počítači
 - veľká efektivita kódu

Vývoj jazyka C

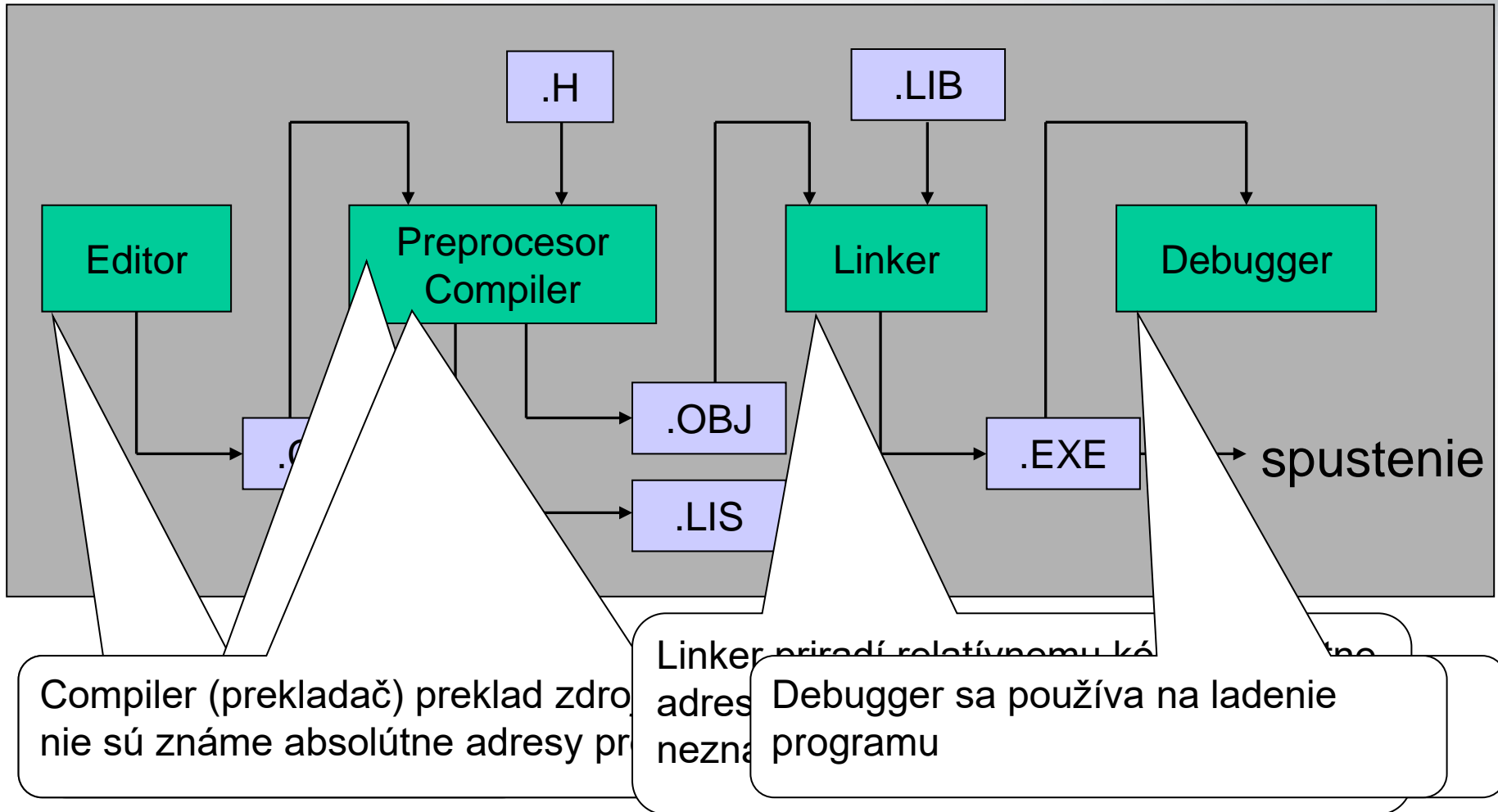


- prvý štandard
 - Kernighan a Ritchie: the C Programming Language v Bell Laboratories (1978) - "vyrástol" z jazyka B
- dnešný štandard:
 - ANSI - 100 %- prenositeľný
(skratka: american national standards institute)

Spôsob spracovania programu



- spracovanie prebieha vo fázach:



Prvý program v jazyku C



- Výpis pozdravu Hello world!

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

umožní používať funkcie na vstup a výstup

```
int main()
```

main je hlavný program

```
{
```

```
    printf(„Hello world!“);
```

Vypíše správu Hello world !

```
    return 0;
```

Main (ako každá iná funkcia) vráti hodnotu

```
}
```

Prvý program v jazyku C, pokračovanie



- Výpis pozdravu *Hello world!*

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

umožní používať funkcie na vstup a výstup

```
int main()
```

```
{
```

```
    printf("Hello world!");
```

```
    return 0;
```

```
}
```

Zdrojové a hlavičkové súbory



- zdrojový program .C
 - je často potrebné doplniť o vložený súbor (knižnicu)
 - jazyk C - nízkej úrovne \Rightarrow nie všetko je súčasťou samotného jazyka, ale definované v knižniciach
- hlavičkové súbory .H
 - zdrojového programu sa .H súbory vkladajú, ak program používa funkcie z nejakej knižnice (napr. funkcie na výpis textu na obrazovku)
 - napr.

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

Prvý program v jazyku C, pokračovanie



- Výpis pozdravu *Hello world!*

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

```
int main()
```

main je hlavný program

```
{
```

```
    printf("Hello world!");
```

```
    return 0;
```

```
}
```

Funkcie



- program pozostáva z funkcií
 - aspoň jedna funkcia: `main`
- viac funkcií:
 - ak je potrebné opakovať nejaký výpočet, vytvorí sa funkcia obsahujúca kód pre tento výpočet - funkcia sa potom volá z inej funkcie (napr. `main`)
 - ak je program príliš dlhý - kvôli prehľadnosti ho rozdelíme do menších častí

```
návratový_typ meno_funkcie(argumenty)
{ telo_funkcie }
```


Prvý program v jazyku C, pokračovanie



Návratový typ: `int` znamená *celočíselný typ*, t.j. funkcia, keď skončí, vráti (povie) nejaké celé číslo

- Výpis programu

Názov funkcie: `main` znamená *hlavnú funkciu* a predstavuje hlavný program

```
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```

Prázdne zátvorky hovoria, že funkcia nemá žiadne *argumenty*. Funkcia (aj hlavná) ale môže argumenty mať (uvidíme neskôr)

Kučeravé zátvorky *uzatvárajú telo funkcie*.

Hlavný program



- funkcia **main**
 - vždy musí byť uvedená v programe
 - funkcia ako každá iná, len je volaná ako prvá pri spustení programu

príklad:

```
int main()  
{  
    ...  
    return 0;  
}
```

Prvý program v jazyku C, pokračovanie



- Výpis pozdravu *Hello world!*

Funcia vracia hodnotu toho typu, ktorú určuje návratový typ: **return** znamená, že *vráti* v tomto prípade *celočíselnú* hodnotu, konkrétne hodnotu 0.

```
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```

Prvý program v jazyku C, pokračovanie



- Výpis pozdravu *Hello world!*

Formátovaný výpis, v úvodzovkách je uzatvorený *formátovací reťazec* - môžu obsahovať všelijaké formátovacie špeciálne znaky

```
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```

Formátovaný výstup



- Výpis premenných podľa formátovacieho reťazca

Formátovací reťazec

```
printf("...", ...)
```

Premenné, ktoré sa vypisujú v rámci textu uzatvorenom vo formátovacom reťazci

Čo sú to premenné? – O chvíľku, len sa ešte pozrime znova a naposledy na program „Hello world!“

Prvý program v jazyku C, pokračovanie



- Vypis pozdravu *Hello world!*

Čo takto, vypísať niečo iné?

```
#include <stdio.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

`\n` - odriadkovanie

Premenné



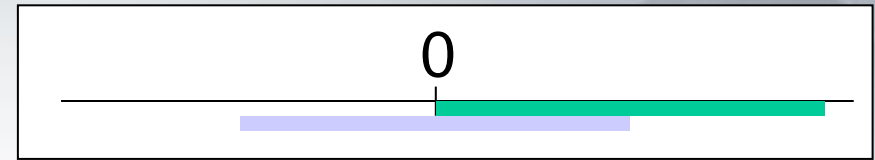
- pomenované pamäťové miesta na ukladanie hodnôt
- hodnoty môžu byť celočíselné, reálne, znakové, alebo reťazcové
- jazyk C je typový - vopred je nutné určiť typ premennej

Jednoduché datové typy



unsigned: rozsah 0 až $2^n - 1$

signed: rozsah -2^{n-1} až 2^{n-1}



- **int** - celé číslo
- **long int (long)** - veľké celé číslo
- **short int (short)** - malé celé číslo
- **char** - znak (znak dosahuje ASCII hodnoty: 0 - 255)
- **float** - reálne číslo
- **double** - väčšie reálne číslo (presnosť 20 desatinných miest)
- **long double** - veľké reálne číslo

Jednoduché dátové typy



vráti počet Bytov typu alebo
premennej

- C zaručuje, že platí:
 - `sizeof(char) = 1 Byte`
 - `sizeof(short int) <= sizeof(int) <= sizeof(long int)`
 - `sizeof(unsigned int) = sizeof(signed int)`
 - `sizeof(float) <= sizeof(double) <= sizeof(long double)`
- C neposkytuje typ `boolean` - booleove hodnoty sa reprezentujú pomocou typu `int`:
 - `FALSE: 0`
 - `TRUE: nenulová hodnota (najčastejšie 1)`

Identifikátory



- jazyk C rozlišuje veľké a malé písmená
 - **pom**, **Pom**, **POM** sú tri rôzne identifikátory
 - kľúčové slová jazyka (**if**, **for**, ...) sa píše s malými písmenami
 - podčiarkovník:
 - **_pom** - systémový identifikátor, nepoužívať
 - **pom_x** - **používať**
 - **pom_** - nepoužívať, často sa prehliadne

Definície premenných



- definícia premennej: príkaz, ktorý priradí premennej určitého typu meno a pamäť
- (deklarácia premennej: príkaz, ktorý len určuje typ premennej, nepriradzuje pamäť → neskôr)

definície:

```
int i;  
char c, ch;  
float f, g;
```

definícia premennej **i** typu **int**

definícia premenných **c**, **ch** typu **char**

definícia premenných **f**, **g** typu **float**

Globálne a lokálne premenné



globálnu premennú môžu používať v celom programe

```
int i;
```

```
int main()
```

```
{
```

```
    int j;
```

```
    ...
```

```
    return 0;
```

```
}
```

kučeravé zátvorky - vymedzujú blok

lokálnu premennú môže používať len v bloku, v ktorom je premenná definovaná

Priradenie



- *l-hodnota* - predstavuje adresu, kam je možné priradiť hodnotu
 - premenná **x** je l-hodnotou
 - konštanta **123** nie je l-hodnotou
- terminológia:
 - výraz: má hodnotu, napr. **i * 2 + 3**
 - priradenie: priradenie hodnoty, napr. **j = i * 2 + 3**
 - príkaz: priradenie ukončené bodkočiarkou,

napr. `j = i * 2 + 3;`

príklady priradení

```
j = 5;  
d = 'z';  
f = f + 3.14 * i;
```

Priradenie – detailnejšie



```
int i;  
  
i = 5;  
  
i = i + 1;  
  
i = i * 5 - 20;
```

- Do premennej na ľavej strane sa priradí hodnota z pravej strany =
- Nejde o rovnicu!

Inicializácia premenných



```
int i = 5;
```

```
i = i + 1;
```

```
i = i * 5 - 20;
```

- Priradenie hodnoty priamo v definícii

Program s premennými



```
int main()
{
    int i, j;

    i = 5;
    j = -1;
    j = j + 2 * i;

    return 0;
}
```

*Definícia celočíselných
premenných
pomenovaných i a j*

Priradenie hodnoty do
premennej i

Priradenie hodnoty do
premennej j

Priradenie hodnoty výrazu
do premennej j

Výpis jednej celočíselnej premennej – použitie príkazu `printf()`



na obrazovku vypíše
hodnotu premennej `i`

```
printf("%d", i);
```

`%d` určuje formát výpisu (dekadické celé číslo)
Prečo nie `%i` ako `int` (celé číslo)? Pretože celé číslo sa dá
vypísať v rôznych sústavách – desiatkovej (dekadickej),
dvojkovej, ...

Program s premennými aj výpisom



```
#include <stdio.h>

int main()
{
    int i, j;

    i = 5;
    printf("%d", i);
    j = -1;
    printf("%d", j);
    j = j + 2 * i;
    printf("%d", j);

    return 0;
}
```

Aké hodnoty sa vypíšu?

Výpis viacerých premenných v rámci jedného `printf()`



```
#include <stdio.h>

int main()
{
    int i, j;

    i = 5;
    j = -1;
    printf("%d %d", i, j);
    j = j + 2 * i;
    printf("%d", j);

    return 0;
}
```

Ako sa hodnoty sa
vypíšu teraz?

Výpis viacerých premenných aj s textom



```
#include <stdio.h>

int main()
{
    int i, j;

    i = 5;
    j = -1;
    printf("i: %d, j: %d", i, j);
    j = j + 2 * i;
    printf("j: %d", j);

    return 0;
}
```

A teraz?

Načítanie celočíselnej premennej – použitie príkazu `scanf()`



```
scanf("%d", &i);
```

`%d` určuje formát čítania
(dekadické celé číslo)

`&` je nutný - znamená adresu premennej,
kam sa má uložiť premenná (vynechanie -
častá chyba)

prečíta celé číslo a
uloží ho do
premennej `i`

Príklad s načítaním premenných



```
#include <stdio.h>

int main()
{
    int i, j;

    scanf("%d", &i);
    scanf("%d", &j);
    printf("i: %d, j: %d\n", i, j);
    j = j + 2 * i;
    printf("%d\n", j);

    return 0;
}
```

Na čo program čaká?
Ako to, že nič nevypisuje?
Čo nakoniec vypíše?

Príklad s načítaním viacerých premenných v jednom scanf ()



```
#include <stdio.h>

int main()
{
    int i, j;

    scanf("%d %d", &i, &j);
    printf("i: %d, j: %d\n", i, j);
    j = j + 2 * i;
    printf("%d\n", j);

    return 0;
}
```

Pri zadávaní hodnôt, okrem medzery môžete použiť aj <Enter>.

Terminálový vstup a výstup



- vstupno/výstupné operácie nie sú časťou jazyka C, obsahuje ich štandardná knižnica
 - dôvodom je, že najviac strojovo závislých akcií je práve vstupno/výstupných - oddeľujú sa nezávislé a strojovo závislé časti jazyka
- popis funkcií na vstup a výstup sa nachádza v hlavičkovom súbore **stdio.h** - pripojíme ho do programu príkazom:

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


Formátovaný vstup a výstup



- funkcie, ktoré načítajú číslo ako reťazec a prevedú ich do číselnej podoby

Pred menom premennej je **&**,
napr. **&i**

- vstup:

```
scanf ("...", ...)
```

"..." - formátovací
reťazec,
... - premenné

- výstup:

```
printf ("...", ...)
```

Formátovací reťazec



- `scanf()` a `printf()` majú premenný počet argumentov
⇒ formátovací reťazec na určenie počtu a typov premenných
- formátovací reťazec obsahuje:
 - formátovacie špecifikácie - začínajú znakom % a určujú formát vstupu alebo výstupu
 - znakové postupnosti - nezačínajú % a vypíšu sa tak, ako sú napísané (používajú sa len v `printf()`)

Formátovaný vstup a výstup: příklad s reálnými čísly



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

umožní používat funkce na vstup a výstup

```
int main()
```

```
{
```

do promenných *i* a *j* načítá číslo

```
    float i, j;
```

```
    scanf("%f %f", &i, &j);
```

vypíše hodnoty
promenných *i*, *j* a
odřadkuje

```
    printf("%f %f\n", i, j);
```

```
    printf("%f je sucet\n", i + j);
```

```
    return 0;
```

vypíše součet hodnot promenných *i*, *j* aj s textom

```
}
```

pre vstup: 2.0, 3.5

vypíše: 2 3.5

5.5 je sucet

Formátovací reťazec



- počet argumentov `scanf()` a `printf()` môže byť väčší ako 2
- počet parametrov musí súhlasiť s formátovacou špecifikáciou
 - počet % = počtu ďalších parametrov
 - ak počty nesúhlasia, kompilátor nehlási chybu, ale program sa nesprávne správa

Načítanie 3 čísel a vypočítanie priemeru



```
#include <stdio.h>

int main()
{
    float k, l, m;

    scanf("%f %f %f", &k, &l, &m);

    printf("Priemer cisel %f, %f a %f je %f\n",
        k, l, m, (k+l+m)/3.0);

    return 0;
}
```

Špecifikácie riadiaceho reťazca



- za znakom %:

c	znak
d	desiatkové číslo typu signed int
ld	desiatkové číslo typu signed long
u	desiatkové číslo typu unsigned int
lu	desiatkové číslo typu unsigned long
f	float (pre <code>printf()</code> tiež double)
Lf	long double
lf	double
x	hexadecimálne číslo malými písmenami
X	hexadecimálne číslo veľkými písmenami
o	osmičkové číslo
s	reťazec
g	číslo tak, ako je to „najprirodzenejšie“ (z general)

pre `printf()` aj
double

L musí byť veľké

niekedy sa nedá použiť aj pre `printf()`

Aritmetické výrazy



- aritmetický výraz ukončený bodkočiarkou sa stáva príkazom, napr.

`i = 2` je výraz s priradením

`i = 2;` je príkaz

- samotná bodkočiarka je tiež príkaz - nazýva sa prázdny príkaz a využije sa v cykloch
- operátory:
 - unárne
 - binárne
 - špeciálne unárne
 - priradovacie

Unárne operátory



- plus (+)
- mínus (-)
- používanie v bežnom význame

príklad:

```
...  
x = +5;  
y = -7;  
...
```


Binárne operátory



- sčítanie (+)
- odčítanie (-)
- násobenie (*)
- reálne delenie (/)
- celočíselné delenie (/)
- modulo (%)

či je delenie celočíselné alebo reálne závisí na type operandov:

int / int → celočíselné
int / float → reálne
float / int → reálne
float / float → reálne

```
int i = 5, j = 13;
```

```
j = j / 4;
```

```
j = i % 3;
```

celočíselné delenie: $13 / 4 = 3$

modulo: zvyšok po delení $5 \% 3 = 2$

Celočíselné a reálne delenie



```
#include <stdio.h>

int main()
{
    int i, j;
    float k, l;

    scanf("%d %d", &i, &j);
    printf("%f", i/j);

    scanf("%f %f", &k, &l);
    printf("%f", k/l);

    return 0;
}
```

Opakovanie



```
int main()
{
    int i;
    float r = 0.25;
    char c1, c2;

    c1 = 'a';
    c1 = c1 + 1;
    c1 = c2 = '\n';

    i = 2;
    r = r * i;
    return 0;
}
```

definovali sme:

- celočíselnú premennú `i`,
- reálnu premennú `r`, ktorú sme inicializovali na 0.25
- premenné pre znak `c1`, `c2`

`c1 = 'a'`

`c1 = 'b'`

`c1 = '\n', aj c2 = '\n'`

`i = 2`

`r = 0.5`

Komentáre



- slúžia na krátke vysvetlenia častí programu, aby sa v ňom vyznal niekto druhý ale aj vy sami
- sprehl'adňujú kód

```
/* komentar */
```

- aj viac riadkov
- C nedovoľuje vhniedzdené komentáre

```
/* komentar v nom /* dalsi komentar */ */
```

Reálne a celočíselné delenie – aj s komentármi



```
/* Realne a celociselne delenie */

#include <stdio.h>

int main()
{
    int i, j;          /* celociselne konstanty */
    float k, l;        /* realne konstanty */

    scanf("%d %d", &i, &j);
    printf("%f", i/j);    /* celociselne delenie */

    scanf("%f %f", &k, &l);
    printf("%f", k/l);    /* realne delenie */

    return 0;
}
```

Formátovaný vstup a špeciálne znaky



- Ako napísať znaky %, \ a pod., keď sú vyhradenými znakmi na formátovanie?

znak	sekvencia
%	%%
\	\\
enter	\n
tabulátor	\t
uvodzovka “	\“

ASCII tabuľka



- znaková sada
 - znakom je priradená hodnota od 0 do 255
 - bežne sa pracuje so znakmi od 0 do 127
 - horná polovica tabuľky - znaky národných abecied

(skratka z: American Standard Code for Information Interchange)

ASCII tabuľka



riadiace znaky	0	-	31
medzera	32	' '	
pomocné znaky	33	'!'	- 47 '/'
čísllice	48	'0'	- 57 '9'
pomocné znaky	58	':'	- 64 '@'
veľké písmená	65	'A'	- 90 'Z'
malé písmená	97	'a'	- 122 'z'
pomocné znaky	123	'{'	- 126 '-'

- neviditeľné znaky:
 - 7 Bell, 8 BackSpace, 9 Tab, 10 LineFeed, 13 Carriage Return,
...

Formátovací řet'azec: příklady



Předpokladajte, že v *i* je hodnota 2 a v *j* je hodnota 3

```
printf("Sucet je: %d", i + j);
```

vypíše: Sucet je: 5

```
printf("Pracovali na 100%%");
```

vypíše: Pracovali na 100%

```
printf("sucet: %d, sucin: %d", i + j, i * j);
```

vypíše: sucet: 5, sucin: 6

Formátovací reťazec: výpis čísel



Výpis na daný počet desatinných miest

```
float pi = 3.1415;  
printf("Pi: %.2f", pi);
```

vypíše: **Pi: 3.14**

Výpis s %g:

```
float pi = 3.1415;  
printf("Pi: %g", pi);
```

Ak nevieme, aké veľké číslo sa bude vypisovať a nie je stanovená presnosť, použijete %g (general). Tu vypíše:
Pi: 3.1415

Formátovací reťazec: příklady



```
printf("\007Chyba, pokus delit nulou");
```

pískne a vypíše: Chyba, pokus delit nulou

```
printf("Toto je \"backslash\": '\\'\n");
```

vypíše: Toto je "backslash": '\ ' a odriadkuje

```
printf("Toto je 'backslash': '\\'\n");
```

vypíše: Toto je 'backslash': '\ ' a odriadkuje

Vstup a výstup znaku



- vstup: `int getchar()`
 - výstup: `void putchar(int c)`
- pracují s promennými typu int
- při volání `getchar()` píšeme znaky pokým nestlačíme <Enter>. Potom přečítá funkce první písmeno a ostatné ignoruje

Vstup a výstup znaku: příklad



program přečte znak z klávesnice, vytlačí ho a odřádkuje

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

```
int main()  
{
```

```
    int c;
```

```
    c = getchar();
```

```
    putchar(c);
```

```
    putchar('\n');
```

```
    return 0;
```

```
}
```

umožní používat funkce na vstup a výstup

načte znak

vypíše načítaný znak

odřádkuje



Chod'te a učte sa programovať!