Pseudokód

- Odsazení označuje blokovou strukturu. Používá se i pro vyznačení těl podmínkových příkazů (***if-else***) a cyklů (***for***, ***while***, ***do-while***).
- Symbol // označuje, že zbývající část řádku následující za tímto symbolem je poznámka.
- Operátory

Pseudokód	Jazyk C
←	=
=	==
≠	!=
≤	<=
≥	>=
mod	%
not	!
and	&&
or	||

- Precedence a asociativita operátorů stejná jako v jazyce C.

Precedence
not unární + a -
* / mod
+ -
< ≤ > ≥
= ≠
and
or
←

Asociativita	
pravě asociativní	levě asociativní
not unární + a - ←	ostatní

- Booleovské operátory ***and*** a ***or*** jsou vyhodnocovány způsobem "short circuit" (obdobně jako v operátory ***&&*** a || v jazyce C).

- Výraz
  - konstanta (číslo) `1` `0.5`
  - proměnná `i` `soucet`
  - složitější výraz sestavený z konstant, proměnných a operací `2*i+1`
- Přiřazení ***a←b←výraz*** přiřadí hodnotu ***výrazu*** oběma proměnným ***a*** a ***b***.
- Úplný podmínkový příkaz

```
if výraz_podmínky
  … příkazy A
else
  … příkazy B
```

- Neúplný podmínkový příkaz

```
if výraz_podmínky
  … příkazy
```

- Parametrický cyklus ***for***

```
for proměnná←výraz₁ to výraz₂
  … příkazy
for proměnná←výraz₁ downto výraz₂
  … příkazy
for proměnná←výraz₁ to výraz₂ step výraz₃
  … příkazy
```

- Cyklus ***while***

```
while podmínkový_výraz
  … příkazy
```

- Cyklus ***do-while***

```
do
  … příkazy
while podmínkový_výraz
```

- Prvek pole označujeme indexem uzavřeným v hranatých závorkách, které jsou za jménem pole. Například ***A[i]*** je prvek pole ***A*** s indexem ***i***. Indexování může být od 1 (první prvek pole je ***A[1]***) nebo od 0 (první prvek pole je ***A[0]***). Protože v současných programovacích jazycích se používá indexování od 0, je vhodnější v pseudokódu rovněž používat indexování od 0.

```
f(A[0..n-1])
f(A,n)
```

- Pokud používáme strukturované datové typy (objekty), členy objektu zapisujeme za jménem objektu oddělené tečkou. Například máme objekt s údaji o osobě *Eva* nazvaný

eva. V něm je (mimo jiné) údaj o datumu narození nazvaný ***datumNarozeni***. Datum narození je sám objektem obsahující členy (proměnné, atributy) nazvané ***den***, ***mesic*** a ***rok***. Pak datum narození objektu ***eva*** zapíšeme jako ***eva.datumNarozeni.den*** .

- Parametry funkcí jsou předávány hodnotou v případě jednoduchých datových typů a odkazem v případě polí a strukturovaných datových typů a u výstupních parametrů.
- Klíčové slovo ***return*** označuje návrat z funkce.
- Návrat z funkce, která vrací hodnotu, je vždy příkazem
  `return` *výraz*.
- U funkcí, jejichž výsledkem je logická hodnota (zjištění, zda něco platí nebo neplatí), se pro logické hodnoty používají označení ***false*** a ***true***.
- Klíčové slovo ***error*** označuje, že byla zjištěna chyba. Řízení je předáno volající funkci, která je zodpovědná za vyřešení chybového stavu (v naší funkci se řešením chybového stavu nezabýváme).
- Odmocnina a mocnina:
  `sqrt(x)` .. druhá odmocnina `x`
  `pow(a,x)` .. mocnina a^x

```
Sqr(x)
  if x≥0
    return sqrt(x)
  error

Sqr(x,y)
  if x≥0
    y ← sqrt(x)
    return
  error
```

```
KvadrFun(a,b,c,x)
  return (a*x + b)*x + c
```

```
Abs(x)
  if x≥0
    return x
  else
    return -x
```

```
Abs(x)
  if x≥0
    return x
  return -x
```

```
Mod(a,b)              // a je celé číslo ≥ 0  b je celé číslo > 0
  return a - a/b*b    // celočíselné dělení

Mod(a,b)
  return a - (a/b)*b
```

```
JeVIntervalu(a,b,x)    // x∈<a,b>
  if a≤x and x≤b
    return true
  return false

JeVIntervalu(a,b,x)
  return a≤x and x≤b
```

```
Soucet(A,n)
  s←0
  for i←0 to n-1
     s ← s + A[i]
  return s

Soucet(A,n)
  s←i←0
  while i<n
    s ← s + A[i]
    i ← i+1
  return s
```

```
Soucet(A,n)
  s←i←0
  do
    s ← s + A[i]
    i ← i+1
  while i<n
  return s
```

```
Faktorial(n)
  if n<2
    return 1     // 0!=1 1!=1
  f←1
  for i←2 to n
    f ← f*i
  return f
```

```
Fibonacci(n)     // 1 1 2 3 5 8 ...
  if n<3
    return 1
  a←b←1
  for i←3 to n     // a b a+b
    x ← a+b
    a ← b
    b ← x
  return x
```