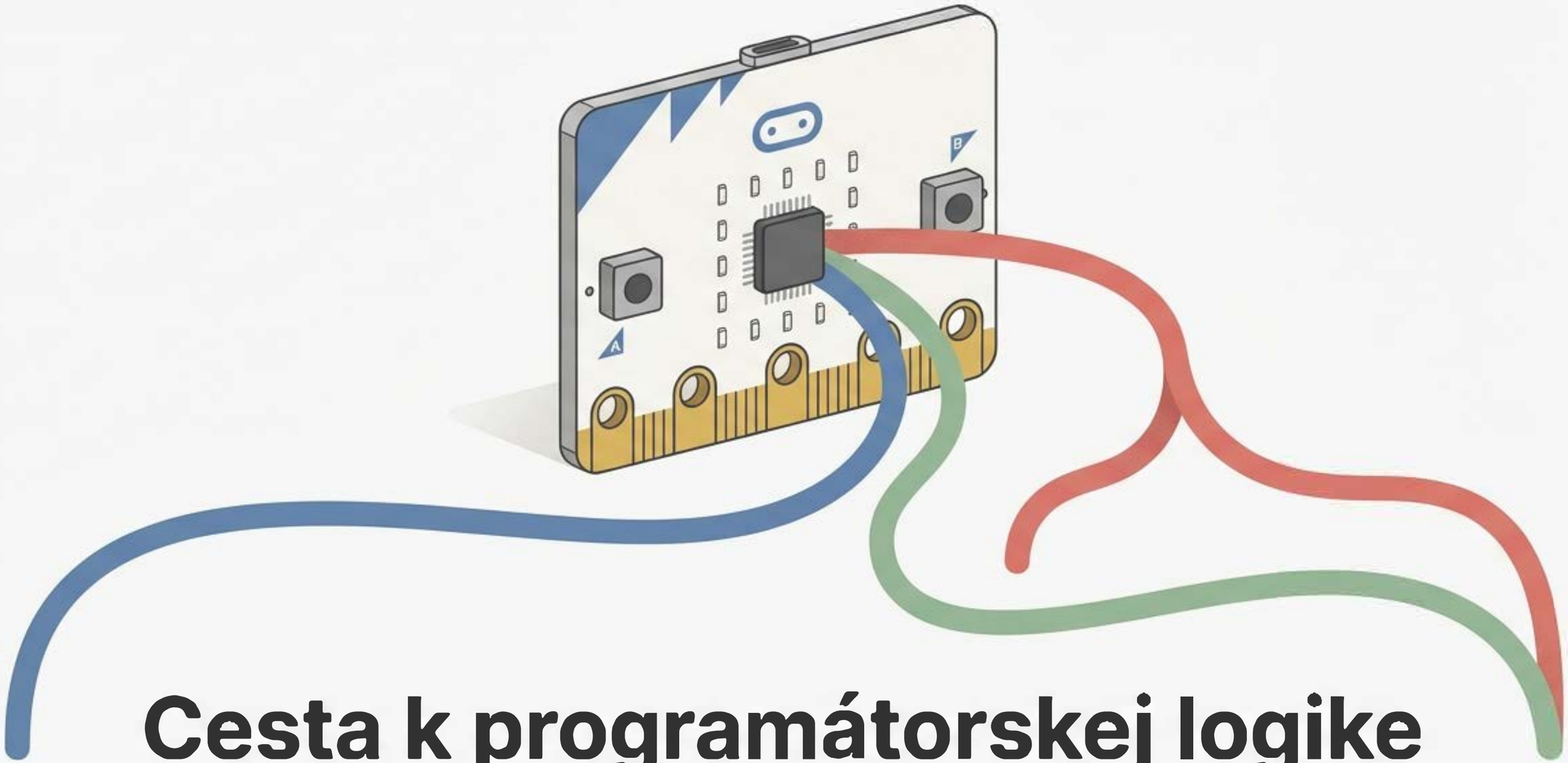
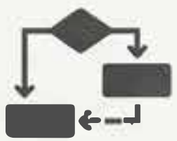


Cesta k programátorskej logike

Naučte svoj micro:bit ‘premýšľať’ pomocou blokov MakeCode

Čo nás čaká na našej ceste?



1. **Prvé rozhodnutie:** Základný príkaz `**ak ... potom**`

2. **Rozcestie:** Pridanie alternatívy s `**inak**`

3. **Inteligentné porovnávanie:** Ako porovnávať hodnoty

4. **Vytvorenie pamäte:** Práca s logickými premennými

5. **Komplexné podmienky:** Spájanie logiky s `**a**` / `**alebo**`

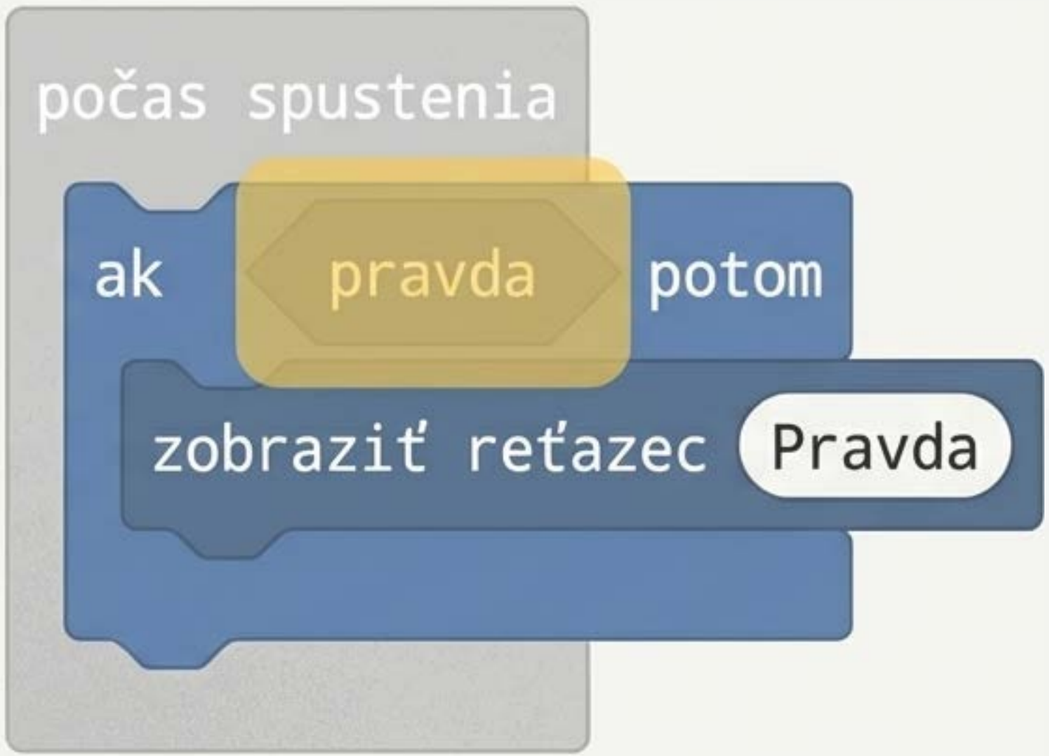
6. **Myslenie v protikladoch:** Použitie negácie `**nie**`

1. Prvé rozhodnutie: Príkaz `ak pravda potom`

Tento príkaz je základom. Umožňuje programu vykonať určitú akciu, ale **iba vtedy**, ak je splnená zadaná podmienka.

Ako to funguje?

1. **Kontrola podmienky:** Program najprv skontroluje, či je podmienka vo vnútri príkazu pravdivá alebo nepravdivá.
2. **Vykonanie kódu:** Ak je podmienka pravdivá, vykoná sa kód umiestnený v bloku `potom`. Ak nieje, program tento blok jednoducho preskočí.

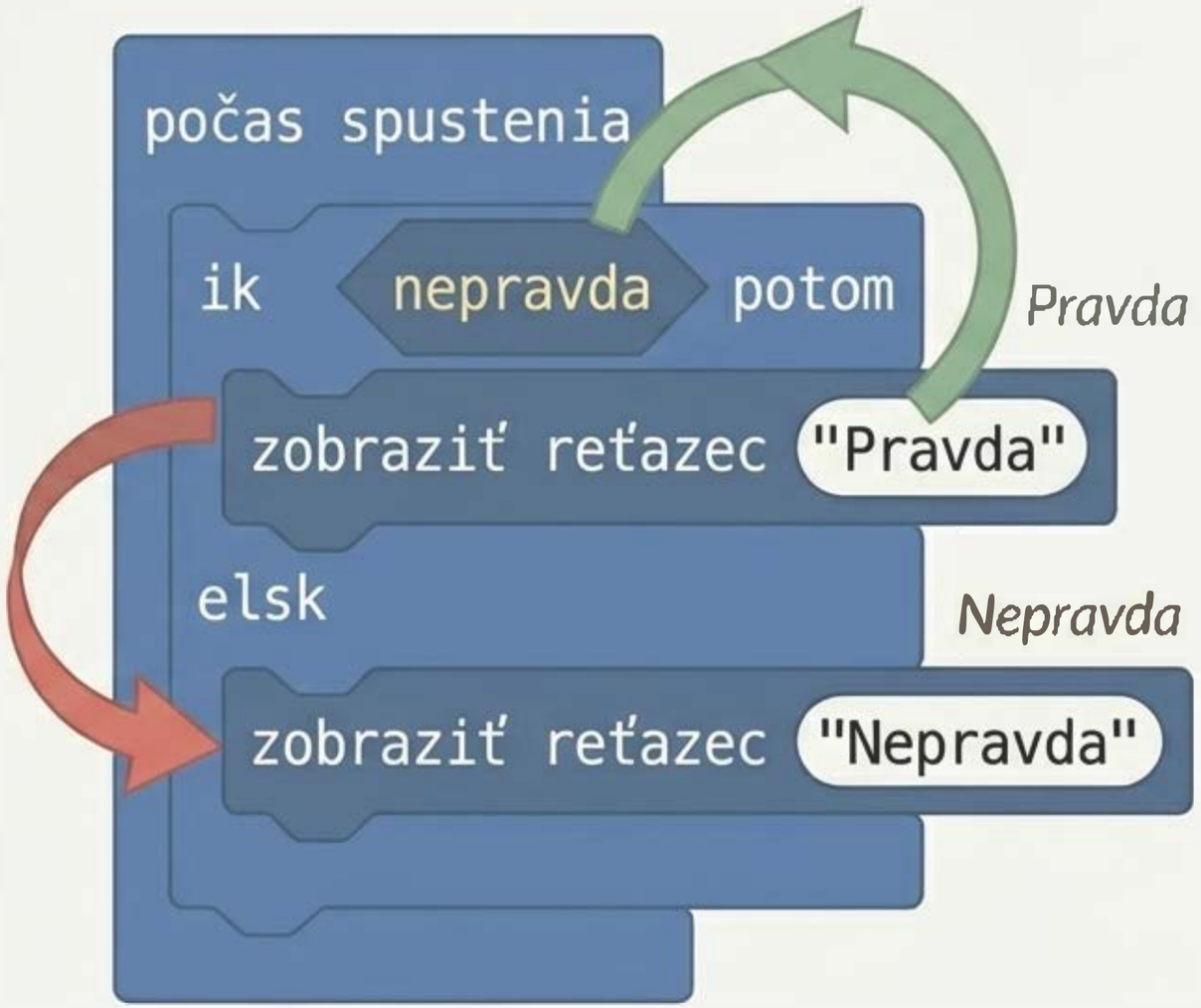


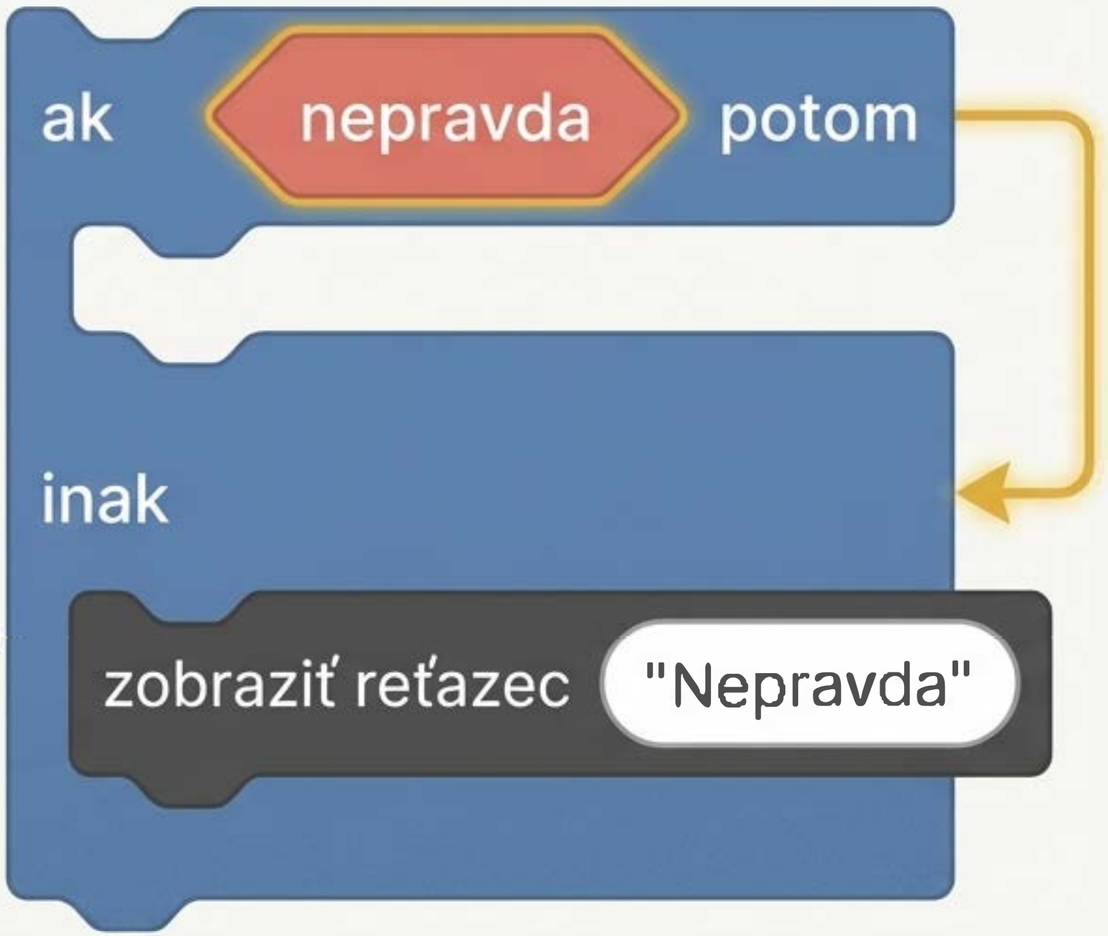
2. Rozcestie: Príkaz `ak pravda potom inak`

Čo ak chceme, aby program niečo urobil, aj keď podmienka **nie je** pravdivá? Príkaz `**inak**` nám dáva alternatívnu cestu.

Ako to funguje?

- **Cesta 1 (Ak je pravda):** Ak je podmienka pravdivá, vykoná sa kód v bloku `potom`.
- **Cesta 2 (Ak je nepravda):** Ak je podmienka nepravdivá, vykoná sa kód v bloku `inak`. Program si vždy vyberie iba jednu z týchto dvoch ciest.





“...potom inak”: Príklad a výzva

Príklad

Podmienka: `nepravda`
Výsledok: Pretože podmienka je nepravdivá, program vykoná kód v bloku `inak` a na displeji zobrazí text “Nepravda”. Keby bola podmienka pravdivá, zobrazil by “Pravda”.

Vyskúšajte si to!

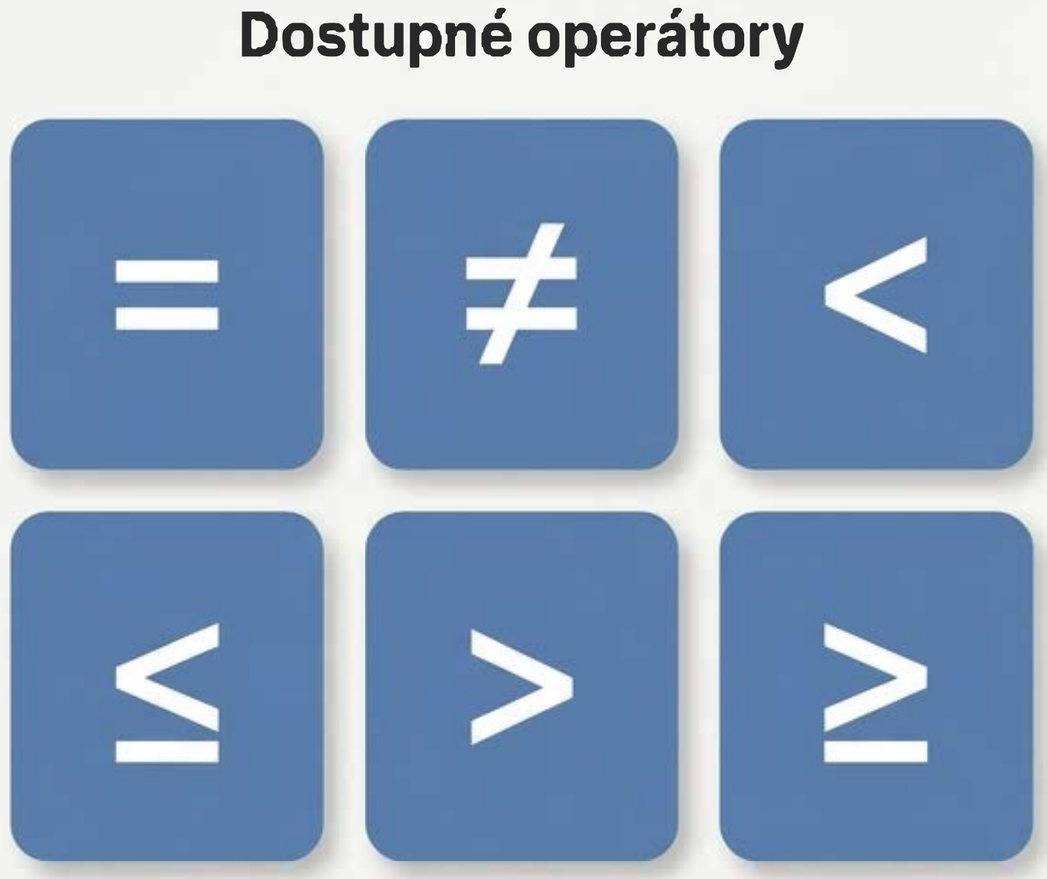
1. Otvorte MakeCode editor.
2. Použite príkaz `**ak pravda potom inak**`.
3. Nastavte podmienku, akciu pre pravdivý prípad a alternatívnu akciu pre nepravdivý prípad.
4. Spustite program a sledujte, ako micro:bit reaguje!

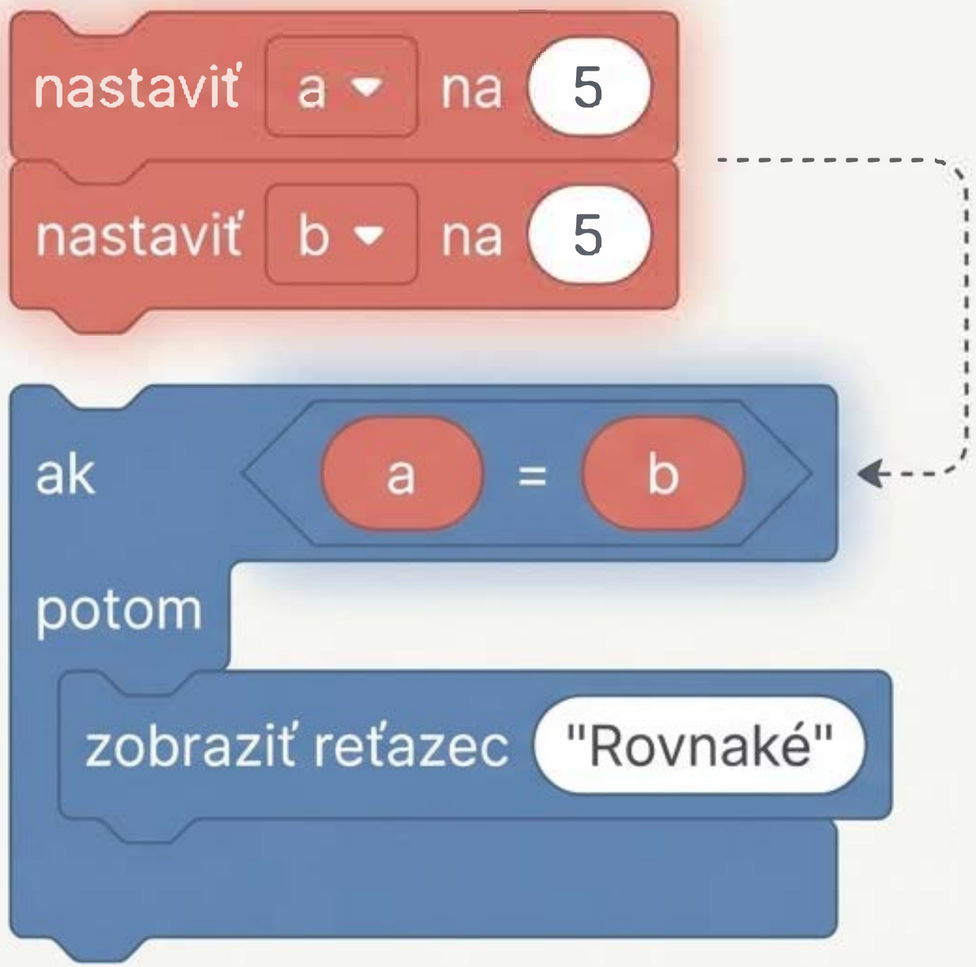
3. Inteligentné porovnávanie: Logika porovnávania

Doteraz sme pracovali s pevne danou “pravdou” alebo “nepravdou”. Skutočná sila logiky spočíva v schopnosti porovnávať hodnoty a na základe toho sa rozhodovať. rozhodovať.

Ako to funguje?

1. **Porovnanie hodnôt:** Do podmienky vložíme porovnávací blok. Ten porovná dve hodnoty (napr. dve premenné) pomocou operátorov.
2. **Vyhodnotenie:** Výsledkom porovnania je vždy buď `pravda` alebo `nepravda`.
3. **Vykonanie kódu:** Ak je výsledok porovnania `pravda`, vykoná sa kód v bloku `potom`.





Porovnávanie v akcii

Príklad

Vytvoríme dve premenné, `a` a `b`, a obom priradíme hodnotu 5.

Podmienka: ak `a = b`

Výsledok: Keďže 5 sa rovná 5, podmienka je pravdivá. Program teda vykoná kód v bloku `potom` a zobrazí text 'Rovnaké'.

Vyskúšajte si to!

1. Vytvorte si dve premenné.
2. Použite príkaz `ak` s porovnávacou logikou.
3. Zmeňte hodnoty premenných alebo operátor (napr. `<` alebo `>`) a sledujte, ako sa zmení správanie programu!

4. Vytvorenie pamäte: Logické premenné

Niekedy potrebujeme v programe uložiť stav – či je niečo zapnuté alebo vypnuté, pravda alebo nepravda. Na to slúžia logické premenné.

Ako to funguje?

1. **Vytvorenie ‘pamäte’:** Vytvoríme premennú (napr. s názvom `stav`), ktorá môže obsahovať iba dve hodnoty: `pravda` (true) alebo `nepravda` (false).
2. **Kontrola stavu:** V podmienke príkazu `ak` sa potom jednoducho opýtame na hodnotu tejto premennej.

Logické premenné v akcii

Príklad

Na začiatku nastavíme logickú premennú `stav` na hodnotu `pravda`.

Podmienka: `ak stav`
Výsledok: Program skontroluje premennú `stav`. Keďže je `pravda`, vykoná sa prvý blok a zobrazí sa text 'Pravda'.

Vyskúšajte si to!

1. Vytvorte si logickú premennú.
2. Použite ju v podmienke príkazu `ak ... inak`.
3. Zmeňte počiatočnú hodnotu premennej z `pravda` na `nepravda`. Čo sa stane, keď `pravda` na `nepravda`. Čo sa stane, keď program spustíte znova?

5. Komplexné podmienky: Spájanie logiky s `a` / `alebo`

Čo ak rozhodnutie závisí od viacerých faktorov naraz? Operátory **`a`** a **`alebo`** nám umožňujú kombinovať viacero podmienok do jednej.

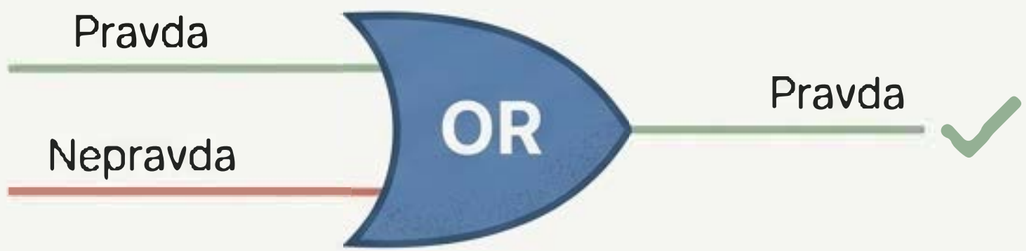
Operátor `a` (AND)

Vyžaduje, aby boli **všetky** spojené podmienky pravdivé. Stačí jedna nepravdivá a celý výsledok je nepravda.

(Pravda **`a`** Pravda) -> **Pravda**
(Pravda **`a`** Nepravda) -> **Nepravda**

Operátor `alebo` (OR)

Vyžaduje, aby bola pravdivá **aspoň jedna** z podmienok. Výsledok je nepravda, iba ak sú všetky podmienky nepravdivé.



(Pravda **`alebo`** Nepravda) -> **Pravda**
(Nepravda **`alebo`** Nepravda) -> **Nepravda**

`a / alebo` v akcii

Nastavíme premennú `a` na `pravda` a `b` na `nepravda`.

1. **Kontrola `ak (a a b)`:** Je pravda A ZÁROVEŇ b pravda? Nie (lebo `b` je nepravda). Tento blok sa preskočí.
2. **Kontrola `inak ak (a alebo b)`:** Je pravda ALEBO b pravda? Áno (lebo `a` je pravda). Vykoná sa tento blok a zobrazí sa 'Jedna pravda'.
3. **Blok `inak`:** Tento blok sa už nevykoná, lebo predchádzajúca podmienka bola splnená. Zobrazil by 'Ani jedna pravda' iba v prípade, že by `a` aj `b` boli nepravda.

Vyskúšaite si to!

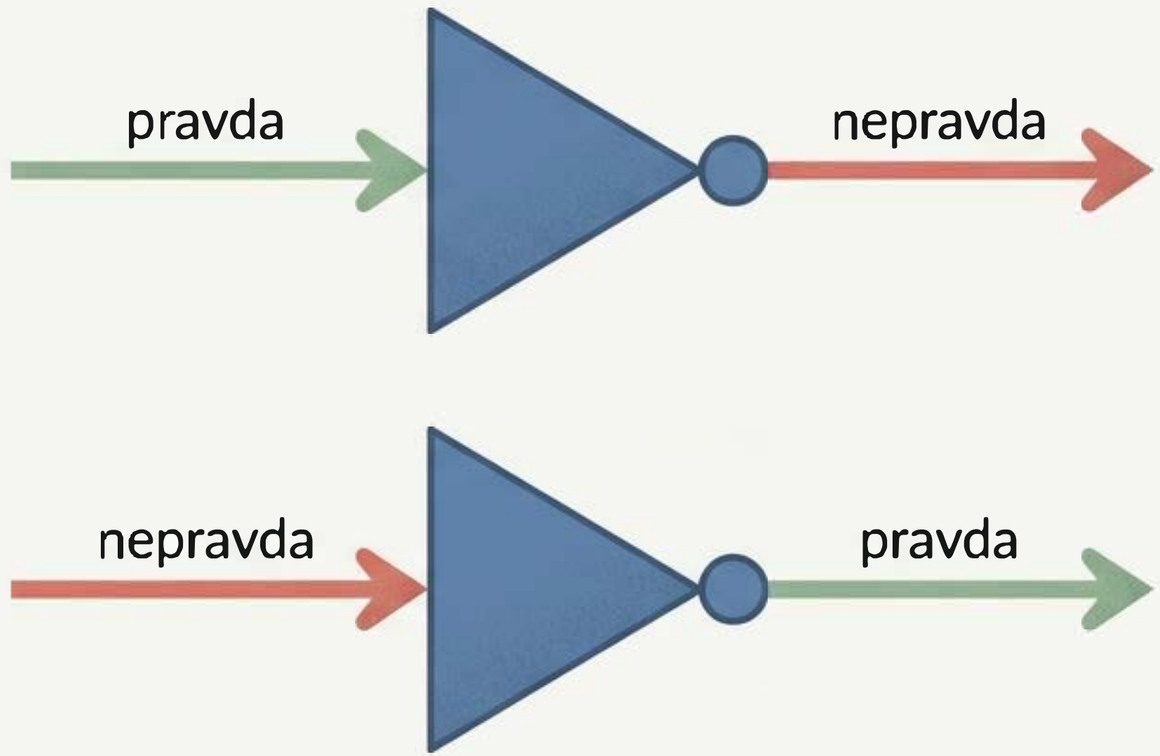
Vyskúšajte všetky štyri kombinácie hodnôt pre `a` a `b` (pravda/pravda', `pravda/nepravda', `nepravda/pravda', `nepravda/nepravda) a sledujte, ktorý text sa zobrazí.

6. Myslenie v protikladoch: Použitie negácie `nie`

Operátor `**nie**` je veľmi jednoduchý: zmení `pravda` na `nepravda` a `nepravda` na `pravda`. Umožňuje nám pýtať sa "ak táto podmienka nie je splnená...".

Ako to funguje?

1. **Negácia hodnoty**: Blok `**nie**` zoberie hodnotu logickej premennej a obráti ju.
2. **Kontrola podmienky**: Príkaz `**ak**` potom pracuje s touto obrátenou hodnotou.



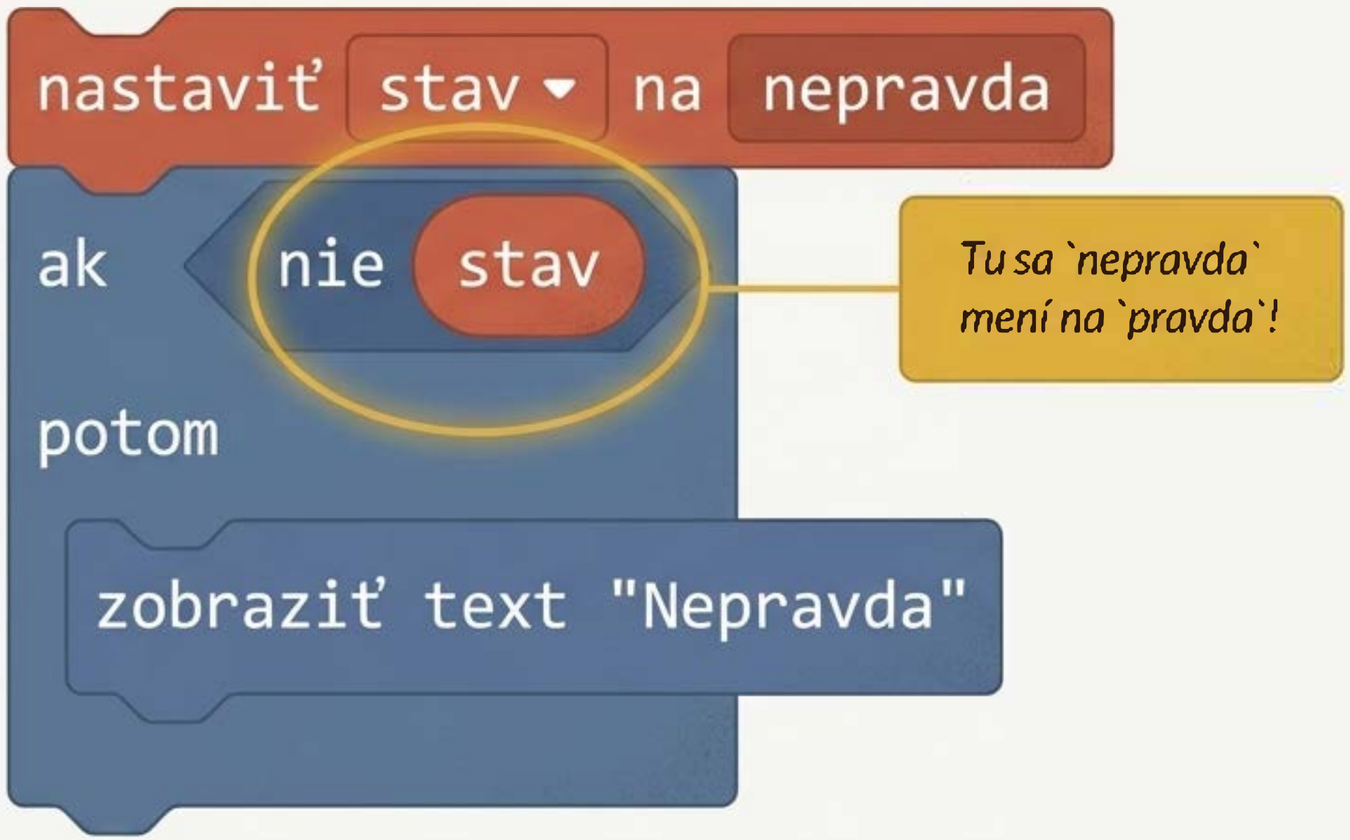
Príklad

Premenná `stav` má hodnotu `false`.

Podmienka `ak nie stav` sa teda prekladá ako `ak nie nepravda`, čo je to isté ako `ak pravda`.

Podmienka je splnená a vykoná sa prvý blok kódu.

Negácia `*nie*` v akcii



Príklad z kódu

Premenná `stav` je nastavená na `false`.
Podmienka: `ak nie stav` sa stáva `ak pravda`.
Výsledok: Keďže je podmienka po negácii pravdivá, micro:bit zobrazí text 'Nepravda' (ktorý je v našom príklade v bloku `potom`).

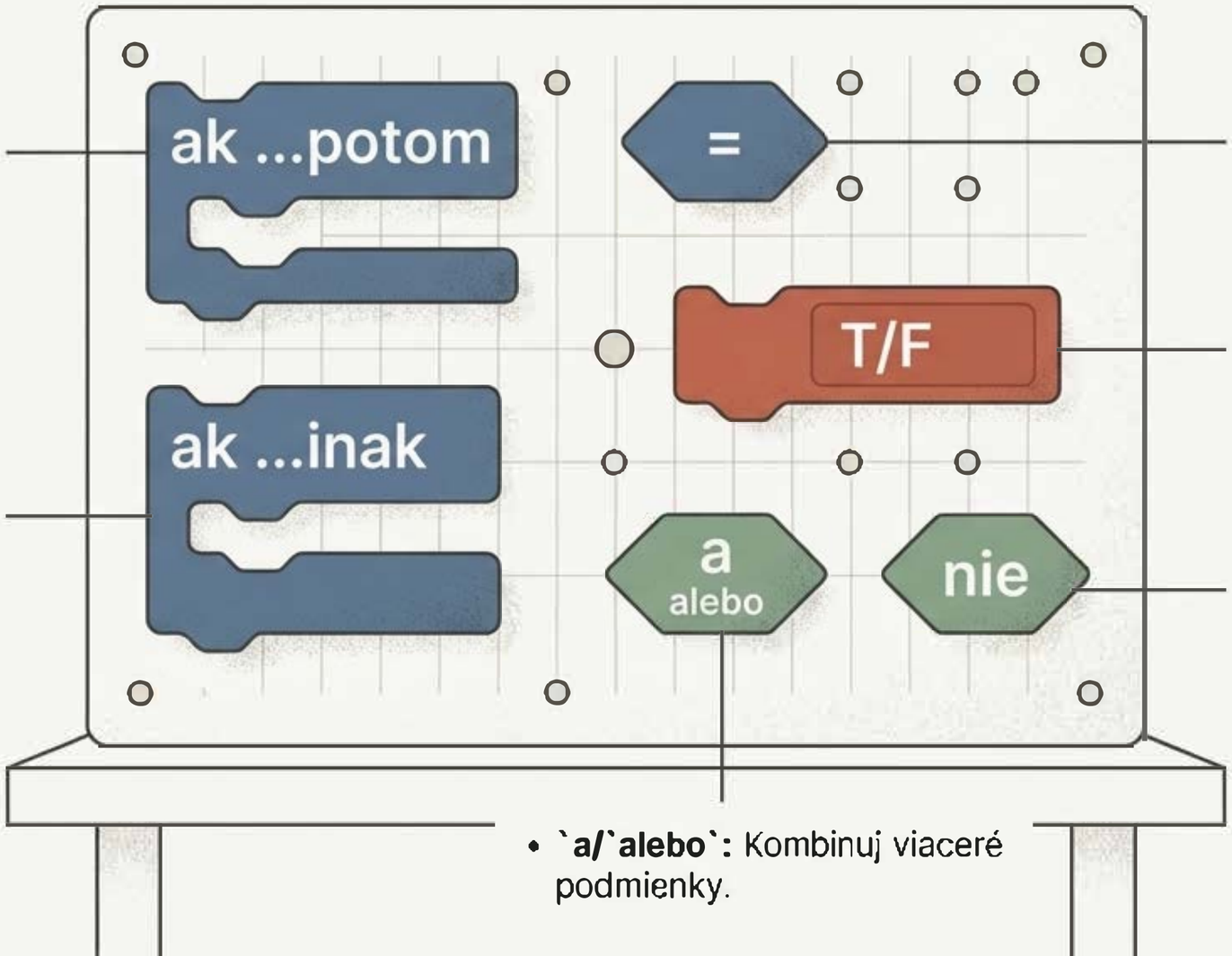
Vyskúšajte si to!

1. Použite príkaz ak s logickou premennou a operátorom `nie`.
2. Nastavte premennú na `true` a sledujte, ako sa program zachová.
3. Skúste operátor `nie` použiť na zložitejšiu podmienku, napríklad `nie (a a b)`.

Váš súbor nástrojov pre programátorskú logiku

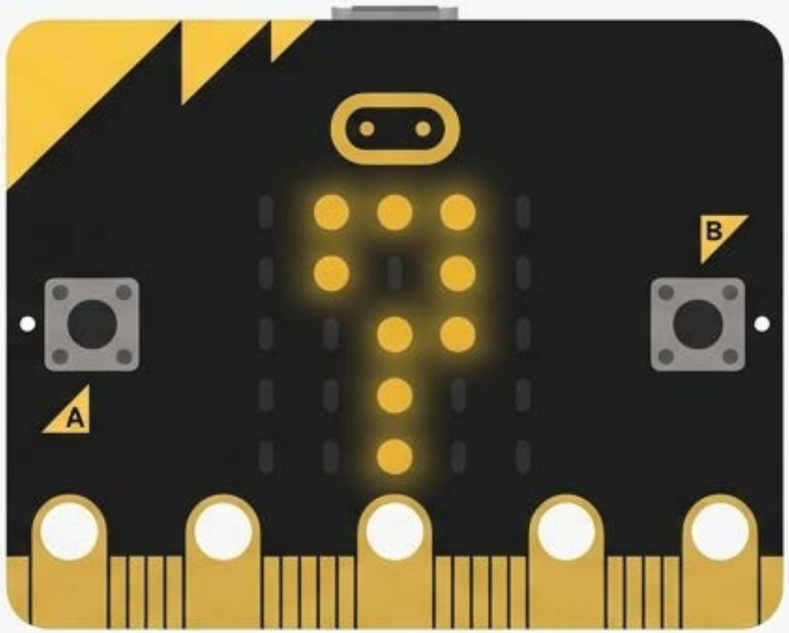
Zvládli ste cestu od jednoduchého rozhodnutia až po komplexné podmienky. Teraz máte k dispozícii kompletnú sadu nástrojov, aby vaše micro:bit projekty reagovali inteligentne na svet okolo seba.

- **`ak ... potom`:** Vykonaj kód, len ak je splnená podmienka.
- **`... inak`:** Poskytni alternatívnu akciu.



- **Porovnávanie (`=`, `>`, `<`...`):** Rob rozhodnutia na základe dát.
- **Premenné (`pravda/nepravda`):** Daj svojmu programu pamäť.
- **`nie`:** Obráť logiku naruby.
- **`a/`alebo`:** Kombinuj viaceré podmienky.

Cesta sa tu nekončí.



Logika je jazyk, ktorým komunikujete s technológiami. Teraz, keď ovládate jej základy, môžete vytvárať programy, ktoré reagujú na stlačenie tlačidla, na teplotu, na svetlo alebo na správy od iných micro:bitov.

Čo inteligentné postavíte ako prvé?