Kapitola 4
Rozhodujeme v podmínkách

V předchozí kapitole jsme mimo jiné mluvili o pravdivostních hodnotách, o Pravdě a Nepravdě, o datovém typu `boolean` a o logických operátorech. Přišli jsme na to, že logické operace nejsou nic tajemného ani složitého, že fungují jako obyčejné sčítání a odčítání. Známe už dokonce i pravidla, podle kterých se logické výrazy zjednodušují. To všechno se nám teď bude hodit. Vše využijeme na skutečných, konkrétních zdrojových kódech.

Využíváme výsledky

Podmínky v programování fungují velmi podobně jako v reálném životě:

- pokud získáš míč, utíkej s ním směrem k bráně,
- když zajde slunce a nastane tma, rozsviť světla,
- jestli chceš kávu z automatu, zaplať nejprve požadovaný obnos,
- jestli chceš jít do divadla, vezmi si oblek,
- pokud chceš jít do bazénu, vezmi si plavky.

Tyto podmínky vypadají vždy stejně: nejprve položíme dotaz, na který existuje jasná odpověď ano, nebo ne. `True`, nebo `False`. Potom přichází rozkazovací způsob (příkaz): něco udělej! Celé schéma bychom mohli zapsat takto:

```
Pokud - Podmínka - Pak - Příkaz.
```

Na schématu je určitě nejdůležitější druhý krok: Podmínka. *Podmínkou* je výraz typu `boolean` s hodnotou True nebo False. Je to vlastně docela jednoduché: jestliže podmínka platí, má hodnotu True. Jestliže podmínka neplatí, má hodnotu False.

Jde o stejný princip jako v předchozí kapitole u výkladu datového typu `boolean`. Nyní ovšem využijeme výsledku – využijeme hodnoty daného výrazu. Co to znamená v lidské řeči: využijeme zkrátka oné hodnoty True, či False k provádění nějakých příkazů. Jak?

Víme, že výraz „Země obíhá kolem Slunce“ je pravdivý. Má hodnotu True. Můžeme tedy napsat: Pokud Země obíhá kolem Slunce, rozsviť v noci světla. Využili jsme hodnoty True, abychom zadali nějaký jasný příkaz. Výše zmíněné schéma tedy můžeme zapsat také takto:

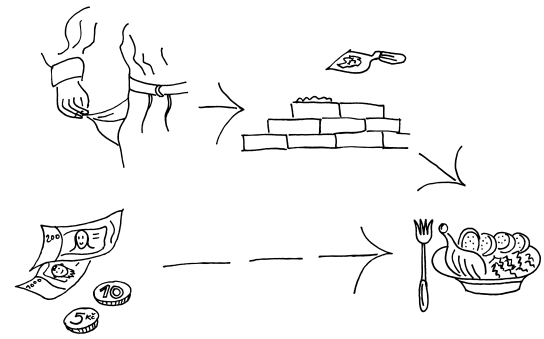
```
Pokud - True - Pak - Příkaz.
Pokud - False - Pak - jiný Příkaz.
```

Poněkud nepříjemný je ten fakt, že výraz „Země obíhá kolem Slunce“ je pravdivý vždy. Takže příkaz by se vždy provedl. My však budeme chtít *zjišťovat stav věcí* a podle toho se

budeme rozhodovat, jaký příkaz provést. Např. se můžeme ptát:

```
Pokud - Zákazník vhodil
dostatečný obnos peněz - Pak
- Vydej mu kávu.
Pokud - Zákazník nevhodil
dostatečný obnos peněz - Pak -
Vypiš na display zprávu.
```

Tato podmínka neplatí vždy, proto potřebujeme zjišťovat *stav věcí*. A po tomto zjištění využijeme výsledku, jak ukazuje obrázek 4.1.



Obrázek 4.1. Pokud mám u sebe peníze, koupím si oběd, pokud peníze nemám, půjdu si je do práce vydělat

Příklady v jazyce Pascal

Vraťme se k vývojovému prostředí jazyka Pascal a vyzkoušejme, jak to funguje při programování. Napíšeme program, který zjišťuje výsledek podmínky a na základě toho provádí nějaký příkaz. Napišme tento kód:

```
program Podminky;
var vek: integer;
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  if vek >= 18 then
    writeln('Jste plnoletý.');
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```

Z dobrého důvodu jsme výše oddělovali všechny čtyři části podmínkového výrazu pomlčkami. Slova Pokud a Pak se v jazyce Pascal píší vždy stejně: `if` (pokud platí) a `then` (potom proveď následující příkaz).

Po předchozí kapitole jsou první řádky kódu jasné. Proto pouze pro úplnost zopakujeme, že do proměnné `vek` typu `integer` (celé číslo) se příkazem `readln` načte hodnota zadaná uživatelem. Tato hodnota je potom hlavním ukazatelem v následující podmínce.

Výraz `vek >= 18` znamená „hodnota proměnné `vek` je větší nebo rovna 18“. Důležitý je samozřejmě i datový typ proměnné. Můžeme testovat rovnost k celému číslu, protože proměnná vek je typu celé číslo. Pokud bychom deklarovali proměnnou jako řetězec (datový typ `String`), nahlásil by překladač chybu na řádku podmínky: takto nelze testovat velikost řetězce vzhledem k celému číslu.

Pokud tedy platí, že hodnota proměnné vek je větší nebo rovna číslu 18, hodnota tohoto výrazu je True. A celý řádek by vypadal takto: `if - True - then - příkaz`. Pokud je zadaný věk větší nebo roven 18, může se provést příkaz, který na obrazovku vypíše text „Jste plnoletý“.

Poznámka: Příkaz `writeln` může být buď na stejném řádku (oddělený mezerou od klíčového slova `then`), nebo na novém řádku, jak to zde máme my. Funkčnost je v obou případech stejná.

Opačný případ

Pokud je ovšem zadaný věk menší než 18, co se stane pak? Protože jsme nic takového neurčili, nestane se nic. Program prostě pokračuje dál k následujícímu řádku a vypíše „Pro ukončení stiskněte Enter". Aby program něco prováděl i v tomto opačném případě, museli bychom to specifikovat. Například takto:

```
if vek < 18 then
  writeln('Nejste plnoletý, nemůžete pít alkohol.');
```

Podmínku musíme zapsat ještě před příkaz `writeln`, který mluví o ukončování, takže celý zdrojový kód by měl vypadat takto:

```
program Podminky1;
var vek: integer;
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  if vek >= 18 then
    writeln('Jste plnoletý.');
  if vek < 18 then
    writeln('Nejste plnoletý, nemůžete pít alkohol.');
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```

Elegantní řešení dvou možností

V tomto příkladu máme přesně dvě možnosti: buď výraz platí, nebo neplatí. Buď je uživatel plnoletý, nebo není. True, nebo False, nic jiného. Abychom nemuseli vypisovat dva výrazy (pokud výraz platí, potom příkaz; pokud výraz neplatí, potom jiný příkaz), můžeme říci i tohle:

```
Pokud - výraz platí
- Potom - příkaz
Jinak - jiný příkaz.
```

Slovo jinak se v programovacím jazyce Pascal řekne stejně jako v angličtině: `else`. Vložíme nyní slovo `else` do našeho zdrojového kódu. Bude pak vypadat takto:

```
program Podminky2;
var vek: integer;
```

Obrázek 4.2. Existují jen dvě možnosti: buď pít alkohol můžeme, nebo nemůžeme.

```
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  if vek >= 18 then
    writeln('Jste plnoletý.')
  else
    writeln('Nejste plnoletý, nemůžete pít alkohol.');
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```

Všimněme si zejména toho, že před klíčovým slovem `else` není středník. Jinak ovšem `else` zastupuje celý řádek `if vek < 18 then`. Všimněme si také toho, že výraz `vek < 18` je přesnou negací výrazu `vek >= 18`. Jsou totiž jen dvě možnosti: buď je věk větší nebo roven 18, nebo je věk menší než 18. Jiná možnost není, proto můžeme použít slovo `else`.

Více příkazů u jedné podmínky

Pokud bychom chtěli za podmínkovým výrazem zadat více než jeden příkaz, musíme to překladači sdělit. Jinak si bude myslet, že k podmínce patří jen první zadaný příkaz. Jak na to: všechny příkazy, které patří k jedné podmínce, uzavřeme mezi klíčová slova `begin` a `end`.

```
program Podminky3;
var vek: integer;
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  if vek >= 18 then
  begin
    writeln('Jste plnoletý.');
    writeln('Můžete pít alkohol veřejně.');
  end
  else
  begin
    writeln('Nejste plnoletý.');
    writeln('Pijte alkohol jen tajně nebo vůbec.');
  end;
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```

Opět si můžeme všimnout, že před `else` není žádný středník. Snadno si domyslíme, jak by to dopadlo, kdybychom dva příkazy `writeln` neuzavřeli mezi `begin` a `end`. Překladač by si myslel, že druhý příkaz `writeln` už nepatří k oné podmínce a provedl by ho v každém případě. Pokud by uživatel zadal věk 17, výpis na obrazovku bude vypadat takto:

```
Můžete pít alkohol veřejně.
Nejste plnoletý.
Pijte alkohol jen tajně nebo vůbec.
Pro ukončení stiskněte Enter.
```

To je pochopitelně nesmysl, první a třetí řádek se zcela popírají. Je proto nutné více příkazů za podmínkou uzavírat mezi klíčová slova `begin` a `end`.

Vnořený if – jde to i bez něho

Co když se potřebujeme zeptat na dvě věci? Abychom zjistili, jestli uživatel může řídit auto, potřebujeme vědět:

- jestli má řidičský průkaz,
- jestli právě nepožil alkohol – jestli je střízlivý.

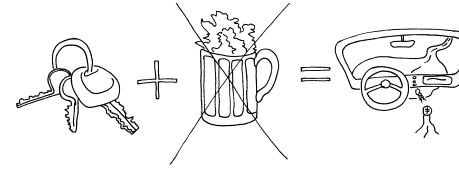
Jak to uděláme: mohli bychom využít vnoření dvou podmínkových výrazů a zeptat se takto:

```
Pokud máte řidičské oprávnění, potom
  Pokud jste střízlivý, potom
    Můžete řídit.
```

Tedy: pokud platí první podmínka, potom – očekáváme příkaz, ale místo něj je tu další podmínka. A pokud platí i tato druhá podmínka, potom teprve vypiš na obrazovku, že uživatel může řídit. Stačí, aby alespoň jedna podmínka neplatila a text „Můžete řídit" se logicky nemůže vypsat.

Ale tohle už přece odněkud známe? Platnost celého tvrzení dává výsledek True, pokud obě obsažená tvrzení dávají také výsledek True. Jde samozřejmě o logický operátor AND známý z předchozí kapitoly. Můžeme tedy obě výše uvedená tvrzení spojit do jednoho a kód bude pracovat úplně stejně jako v předchozím případě:

```
Pokud (máte řidičský průkaz) AND
(jste střízlivý) Potom
Můžete řídit
```



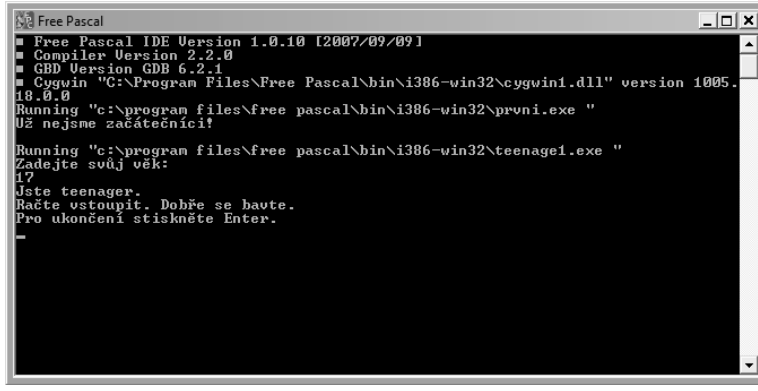
Obrázek 4.3. Schéma operátoru AND

Poznámka: Naučme se používat logické operátory místo složitého vnořování! Celý zdrojový kód se stane jednodušším a přehlednějším.

Než se přesuneme k ostatním operátorům, ukažme si jeden příklad v jazyce Pascal pro již zmíněný operátor AND. Program zjišťuje, jestli můžeme přijít na party „pouze pro teenagery", tedy pokud nám je mezi 13 a 19 lety (anglicky thir**teen** – nine**teen**):

```
program TeenAgeParty;
{Party pro teenagery}
var vek: integer;
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  if (vek > 12) AND (vek < 20) then
  begin
    writeln('Jste teenager.');
    writeln('Račte vstoupit. Dobře se bavte.');
  end;
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```

Jednotlivé výrazy je nutné uzavírat do závorek, aby operátor AND věděl, odkud kam je jeden výraz a kde začíná a kde přesně končí druhý výraz. Závorky to přesně ukážou. Pokud program funguje správně, a není důvod, aby nefungoval, zobrazí se po spuštění a po zadání věku okno jako na obrázku 4.4.



Obrázek 4.4. Výpis na obrazovce po zadání věku teenagera

Přísná logika

Výsledky výrazů propojených logickými operátory se řídí logikou, kterou už známe z předchozí kapitoly. Spojením několika (i mnoha) výrazů nakonec vždy vzniká jediná hodnota. Ta je rovna buď True, nebo False. Takže celý podmíněný příkaz `if - (složený)výraz - then - příkaz(y)` má jasnou úlohu: vyhodnotit složené výrazy a přiřknout jim jedinou Pravdu, nebo jedinou Nepravdu. Poté se již chovají stejně, jak kdyby byl mezi klíčová slova `if` a `then` umístěný jen jediný výraz.

Lze si to vyzkoušet na konkrétním příkladu. Je třeba možné, aby se na party teenagerů dostali výjimečně i oblíbení učitelé Martin (29 let) a Hanka (26 let). Položíme jim tedy kontrolní otázku, na kterou jim předem prozradíme odpověď, aby je dveřní systém při zapsání správné odpovědi vpustil dovnitř.

Poznámka: Otázka by měla být trochu nelogická a téměř nesmyslná, aby se správná odpověď nedala jen tak uhádnout.

```
program TeenAgeParty;
var vek: integer;
    odpoved: String;
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  writeln('Kontrolní otázka:');
  writeln('Jedna květina bez listu,');
  writeln('druhá bez nohy, třetí bez prstu,');
  writeln('kolik to dělá celkem dnů.');
  readln(odpoved);
```

```
  if (vek > 12) and (vek < 20)
                or (odpoved = ('Rovné dva dny'))
    then
    writeln('Můžete vstoupit. Dobře se bavte.')
  else
  writeln('Nemáte oprávnění vstoupit na tuto party.');
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```

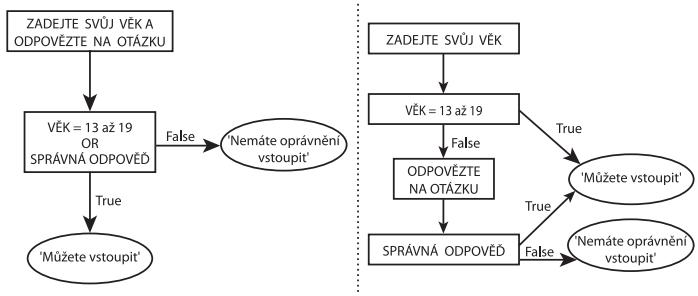
Kontroluje se i velikost písmen, takže pokud napíšeme správnou odpověď, ale s malým začátečním písmenem, dveřní systém nás odmítne.

Poznámka: Kontrolní otázku i její odpověď jsme převzali z oblíbeného večerníčku Mach a Šebestová, konkrétně z dílu Cizí planeta. Část tohoto dílu můžeme zhlédnout např. zde: http://www.youtube.com/watch?v=Ly0fgy0QwXw.

Zdrojový kód pro dveřní systém, který jsme právě napsali, má jednu nevýhodu. Kontrolní otázka se pokládá a vyhodnocuje vždy, tzn. i v případě, že chce vstoupit teenager. Lepší by bylo, kdyby systém pokládal otázku jen těm, kteří mezi teenagery nepatří. Jinak se totiž program ptá zbytečně i těch, kteří by mohli bez problémů vstoupit.

Této zbytečné a nešikovné složitosti jsme se dopustili vědomě. Šlo nám o demonstraci složeného výrazu a jeho vyhodnocení. Aby se otázka pokládala jen těm, kteří nespadají do věku teenagerů, upravíme zdrojový kód tak, že rozdělíme příchozí na dvě skupiny: teenagery vpustíme bez okolků dovnitř na party, ať se baví. Všem ostatním položíme kontrolní zapeklitou otázku:

```
program TeenAgeParty2;
var vek: integer;
    odpoved: String;
begin
  writeln('Zadejte svůj věk:');
  readln(vek);
  if (vek > 12) and (vek < 20) then
    writeln('Můžete vstoupit. Dobře se bavte.')
  else
  begin
    writeln('Kontrolní otázka:');
    writeln('Jedna květina bez listu,');
    writeln('druha bez nohy, třetí bez prstu,');
    writeln('kolik to dělá celkem dnů.');
    readln(odpoved);
    if odpoved = ('Rovné dva dny') then
      writeln('Můžete vstoupit. Dobře se bavte.')
    else
      writeln('Nejste oprávněn vstoupit na tuto party.');
  end;
  writeln('Pro ukončení stiskněte Enter.');
  readln();
end.
```



Obrázek 4.5. Rozdíl dvou schémat: schéma pro zadávání kontrolní otázky všem příchozím a schéma pro zadávání otázky jen těm, kteří nesplňují požadavek věku teenagera

Nyní už se dveřní systém ptá jen těch, kteří nemají věk teenagerů, jak ukazuje obrázek 4.5. Vraťme se však ke složenému výrazu a všimněme si závorek. Závorky nyní oddělují jen samotné jednoduché výrazy, aby logický operátor věděl, kde začínají a kde končí. Nyní tedy vypadá složený výraz takto:

```
if
(vek > 12) and (vek < 20) or (odpoved = ('Rovné dva dny'))
then
```

Můžeme se přesvědčit, že další závorky jsou v tomto případě zbytečné. Oba tyto výrazy dopadnou při vyhodnocování stejně a vpustí dovnitř jen ty, kteří dovnitř smějí (ve výčtu je naznačeno rozložení závorek i rozložení na řádcích):

- `(vek > 12) and`
  `((vek < 20) or (odpoved = ('Rovné dva dny'))),`
- `((vek > 12) and (vek < 20)) or`
  `(odpoved = ('Rovné dva dny')).`

Pokud by však byla podmínka vpuštění složitější, dávejme si dobrý pozor právě na závorky. Kdyby záleželo i na jméně, vypadal by výraz takto:

```
if
(vek > 12) and (vek < 20) or
(odpoved = ('Rovné dva dny')) and
(jmeno=Hanka) or (jmeno=Martin)
then
```

Pokud bychom uzavřeli do závorek vše kromě posledního výrazu, `(jmeno=Martin)`, stačilo by ke vstupu zadat jméno Martin a věk nevěk, odpověď neodpověď by se dveřní systém otevřel.

Tip: Zkusme si dopsat do zdrojového kódu dotaz na jméno a zkusme ho také zařadit mezi podmínky vpuštění. Ovšem se správným uzávorkováním, tedy tak, aby byla podmínka položena logicky.

Výběr z několika možností

Doposud jsme před sebou měli dvě možnosti: buď a nebo. Buď se dveře otevřou, nebo neotevřou, buď řídit můžete, nebo nemůžete. Jenže v životě to bohužel není tak jednoduché,

často nelze věci rozlišit na černé a bílé, na buď a nebo. Někdy je ve hře celá řada možností, a proto v jazyce Pascal existuje příkaz `case`.

Slovo `case` znamená anglicky případ a podle toho se používá a chová i v programu. Příklad, který nás napadne hned jako první je samozřejmě nádražní automat: v případě, že cestující stiskl tlačítko pro kávu, dej mu kávu, v případě, že stiskl čaj, dej mu čaj, v případě stisknutí tlačítka horká čokoláda, vydej horkou čokoládu atd. Pokud bychom převedli příklad do jazyka Pascal, vypadal by kód takto:

```
case Tlacitko of
  1:
    begin
      vydej kávu;
      writeln('Káva je hotová.');
      writeln('Dobrou chuť.');
    end;
  2:
    begin
      vydej čaj;
      writeln('Čaj je hotový.');
      writeln('Dobrou chuť.');
    end;
  3:
    begin
      vydej horkou čokoládu;
      writeln('Čokoláda je hotová.');
      writeln('Dobrou chuť.');
    end;
end;
```

Tento kód by se dal přeložit do češtiny takto:

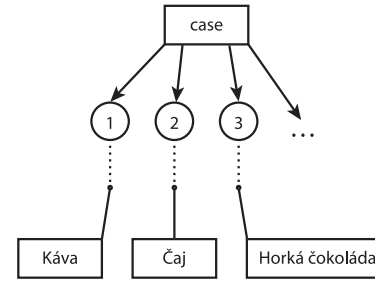
- V případě, že uživatel stiskl číslo 1, vydej kávu.
- V případě, že uživatel stiskl číslo 2, vydej čaj.
- V případě, že uživatel stiskl číslo 3, vydej horkou čokoládu.
- Ve všech případech nezapomeň popřát dobrou chuť.

U příkazu `case` bohužel nemůžeme před dvojtečku zadat přímo slova „káva“, „čaj“ apod., protože jednotlivé možnosti musí být buď typu `integer`, nebo `char`. Tedy mohou zde být buď celá čísla nebo jednotlivé znaky. To pro nás ale není problém. Přiřadíme prostě jednotlivým tlačítkům pořadová čísla a s těmi budeme operovat.

Důležité: Příkaz `case` může pracovat pouze s tzv. ordinálními typy. „Ordinal“ znamená anglicky pořadové číslo, takže ordinální typy jsou takové, které vyjadřují nějaké pořadí. Přesněji jde o takové typy, které jsou omezené, konečné a pro každou hodnotu (s výjimkou krajních hodnot) je jednoznačně určen následník a předchůdce. Typ `integer` to samozřejmě splňuje. Předchůdce čísla 4 je číslo 3, následovníkem je číslo 5.

Datový typ `char` je ale typu `integer` velmi podobný, protože každý znak má své pořadové číslo v ASCII tabulce. Takže jednotlivé znaky lze v příkazu `case` používat také. Typy `String` nebo `real` nemají jednoznačně dané předchůdce a následovníky, takže je musíme nahrazovat celými čísly nebo znaky.

Pokud by za dvojtečkou následoval jen jeden příkaz, nemuseli bychom ho uzavírat mezi klíčová slova `begin` a `end`. Více příkazů, jak vidíme zde, však uzavřít musíme. Zatím sice nemáme automat, který by opravdu vydával horké nápoje, ale můžeme alespoň simulovat jeho textové výstupy. Buďme důslední a napišme si program na výdej horkých nápojů. Napišme do vývojového prostředí tento kód:



Obrázek 4.6. Přiřazení pořadových čísel jednotlivým tlačítkům na automatu je pro využití příkazu case nutné

```
program Automat;
var Tlacitko: integer;
begin
  writeln('Dobrý den, vyberte si nápoj.');
  writeln('1: Káva.');
  writeln('2: Čaj.');
  writeln('3: Horká čokoláda.');
  readln(Tlacitko);
  case Tlacitko of
    1:
      begin
        writeln('Káva je hotová.');
        writeln('Dobrou chuť.');
      end;
    2:
      begin
        writeln('Čaj je hotový.');
        writeln('Dobrou chuť.');
      end;
    3:
      begin
        writeln('Čokoláda je hotová.');
        writeln('Dobrou chuť.');
      end;
    else
      writeln('Pro výdej nápoje stiskněte 1-3.');
  end;
  writeln('Pokračujte stisknutím klávesy Enter.');
  readln();
end.
```

Po stisknutí kláves 1–3 se program chová správně. Pokud stiskneme jiné číslo, program správně ohlásí, že jsme se netrefili do tlačítek, nápoj je pouze v rozmezí čísel 1 až 3. A protože je proměnná `Tlacitko` typu `integer`, celý program by skončil chybou, pokud bychom nechali načíst např. písmeno „d“, nebo slovo „káva“, zkrátka něco mimo typ `integer`. Je to stejné, jako když někdo na nádraží nemačká tlačítka 1–3, ale začne třeba do automatu bouchat. Tím se může automat jedině porouchat, ale kávu tím nikdy nevydá.