Algoritmizace

Algoritmus

Algoritmus je přesný popis pracovního procesu, který z měnitelných vstupních údajů dochází k žádaným výsledkům.

Vlastnosti algoritmus

- **Determinovanost** - v každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat, pro stejné vstupní data musí mít stejný výstup
- **Obecnost** - algoritmus by neměl řešit jeden konkrétní problém (například 5 x 5), ale měl by nabízet obecné řešení daného problému (například X x Y)
- **Finitivnost** - algoritmus by měl vždy mít omezený počet kroků, po kterých skončí
- **Resultativnost** - musí mít nějaký výstup
- **Korektnost** - výstup by měl být správně
- **Efektivita** - dělái se na paměťovou efektivitu (náročnost na paměť) a výpočetní efektivitu (náročnost na výpočet), tyto dvě vlastnosti jsou většinou k sobě ve vztahu nepřímé úměry

Základní prvky algoritmů

Podmínky (selekce)

Umožňují větvení algoritmů. Podmínky můžou mít čtyři možné formy. První forma jsou podmínky typu **if-then**, když něco, tak dělej („Pokud máš řidičák, budeš řídit.“).

Druhou formou jsou podmínky typu **if-else** když něco, tak dělej, a pokud ne něco, tak dělej něco jiného („Když máte kuřecí kůžičky, přidejte je, pokud je nemáte použijte potravinovou fólii.“ - převzato z kuchařky Ládi Hrušky). Podmínka typu **if-else** může mít speciální variantu a tou je tzv. ternární operátor. Ternární operátor se používá ve chvíli, kdy se při splnění podmínky použije jedna hodnota a při jejím nesplnění hodnota druhá (zápis vypadá takto: podmínka ? hodnota při splnění : hodnota při nesplnění).

Existuje ještě jeden typ podmínky a tou je **switch-case**. Tato podmínka je založena na tom, že konkrétní proměnná (například stav) může nabývat několik specifických hodnot a děj algoritmu se ubíhá podle hodnoty, kterou má tato proměnná (Pokud stav nabývá hodnoty čekám čekej, pokud nabývá hodnoty načítám načítej,...).

Cykly

Umožňují vícenásobné opakování části algoritmu. Cykly můžou mít také dvě možné formy. První forma jsou cykly typu **while**, dokud něco tak prováděj („Dokud máš v košíku nákup, skládej ho na pult.“). Druhá forma jsou cykly typu **do-while** prováděj dokud něco(„Ohřívej vodu, dokud nezačne vařit.“). Hlavní rozdíl mezi cykly typu while a do-while je ten, že cyklus typu do-while proběhne

minimálně jednou, zatímco cyklus while pokud nebude hned na počátku splněna podmínka nemusí.

Cyklus typu **for** je speciálním případem cyklu **while**. Cyklus for je založen na tom, že máme číselnou proměnnou, kterou po každém průchodu cyklu upravíme a na jeho začátku kontrolujeme podmínku (Příkladem může být výpočet faktoriálu - „Dokud číslo není jedna, odečti od něj jedničku a vynásob číslem předchozím“). Cyklus **for** se nejčastěji používá pro iteraci přes všechny prvky množiny, kvůli zjednodušení této operace byl ve většině programovacích přidán cyklus **for-each**, který toto umožňuje bez nutnosti pamatování si v kterém místě v rámci množiny jsem.

Ukázka algoritmu

Vajíčka

- **Vstupní údaje:** počet vajec, typ tuku, šunka
- **Výstup:** poživatelná volská oka

1. Vezmi pánev
2. POKUD je typ tuku máslo, vezmi z lednice máslo
3. POKUD NENÍ, vem ze skříně olej (**if-else**)
4. Dej olej na pánev
5. Přidej na pánev šunku
6. DOKUD není šunka dozlatova, čekej (**while**)
7. Přidej vejce, DOKUD tam nejsou všechny (**do-while**)
8. DOKUD vše není hotovo, čekej (**while**)
9. Vypni plotnu
10. Jez!

Pseudokód - klasický kód