Vývojové diagramy

- složí k popisu algoritmu programu
- diagramy jsou různého typu, některé sledují tok programu, jiné tok dat, některé obojí
- nekreslit zbytečně dlouhé a složité struktury – použít podprogramy

algoritmus – popis části programu, může se lišit vstupními daty
Vývojový diagram – grafický popis algoritmu. Součástí může být i slovní popis co se v daných bodech děje (+ seznam ovlivněných či použitých proměnných).

Pro popis algoritmu potřebujeme

- vycházíme z detailního rozboru / popisu úlohy
- definujeme vstupní a výstupní data
- formulujeme algoritmus slovním popisem
- nakreslíme vývojový diagram
- naprogramujeme algoritmus

Algoritmy musí splňovat (ve většině případů)

- dobře definované vstupy a výstupy a činnost
- nesmí se zacyklit bez možnosti opuštění
- nesmí existovat větve programu, které nejsou ošetřeny („nevíme co dělat, když jsou „špatná“ data, tak raděj neděláme nic“).
- musí ošetřit všechny chybové stavy (správnost vstupních dat, možnost operací (dělení nulou ...), kontrolovat správné provedení operací (alokace paměti, otevření souboru ...)
- odevzdat zdroje (naalokovanou paměť, otevřené soubory ...) – pro všechny větve ukončení algoritmu
-

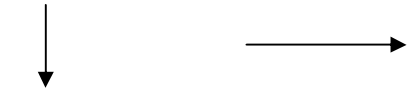
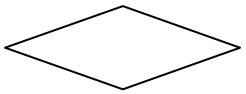
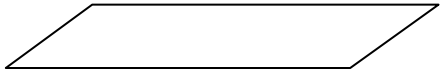
DFD – Data Flow Diagramy

- připomínají stavový diagram
- v uzlech diagramu jsou popsány činnosti, na hranách jsou data
- slouží hlavně k popisu vstupů a výstupů, výměny dat. Ošetření chybných stavů dat.
- Nejprve se zakreslí celkový pohled, potom se provádí zanořování (dělení na menší celky).
- pomáhá při komunikaci se zadavateli neobeznámenými s programováním

Proces – mění vstupy na výstupy (nejčastěji elipsa)
Tok – šíření informace, dat (jednosměrné, nebo obousměrné šipky)
Sklad – uložení zpracovávaných vstupních nebo výstupních dat nebo mezivýsledky čekající na další zpracování – například soubory, databáze (dvě vodorovné čáry)
Terminátor – externí vstup dat nezávislý na popisovaném problému (čtverec)

Nakreslete DFD a (později i) vývojový diagram pro výpočet většího reálného kořene kvadratické rovnice

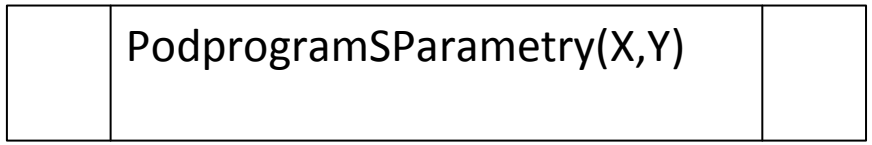
Značky vývojového diagramu

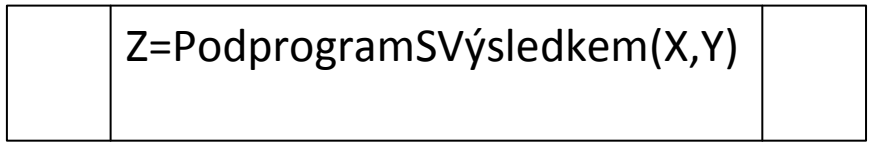
	tok programu, preferované směry
	začátek a konec programu
	příkaz
	podmínka, větvení
	podprogram
	cyklus - for
	vstup nebo výstup dat

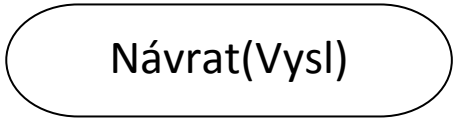
Pro lepší přehlednost je možné použít:

- podprogram
- spojovací body označující stejné místo v diagramu např. na různých listech papíru, nebo pro odstranění dlouhých čar (číslo, písmeno, název v kružnici)
- rozložení příkazu – použije se pojmenovaný blok příkazu, který sestává z více příkazů – rozkreslí se v jiném místě. Použít pro jednodušší celky (například cykly), pro složitější celky využívat podprogram.

Podprogram – volání a realizace (vstupu a výstupu (sub)algoritmu)







Jednu hodnotu vracíme pomocí posledního bloku (return/návrat(x))
Vstupní parametry mohou být vstupní, výstupní, nebo vstupně výstupní (nevyznačuje se v grafu – plyne z kontextu nebo popisu)

Volba (typu) proměnných

- z hlediska typu činnosti – celočíselné (indexace, výpočty s celými čísly), reálné (výpočty v pohyblivé řádové čárce – neceločíselné výpočty), bezznaménkové celočíselné typy (pro logické operace), ukazatele/adresy (pro práci s většími paměťovými celky, rozsáhlejšími daty), struktury (pro práci s proměnnými skládajícími se z většího počtu/souboru proměnných)
- z hlediska přesnosti – char/short/int/long/long long. float/double/long double.
- z hlediska umístění v paměti a viditelnosti – globální (extern), statické globální, statické lokální, automatické, dynamické (alokované za běhu programu)