Vývojové diagramy znázorňují průběh či stavbu programu. Používají se jako část dokumentace projektu. Většina projektů začíná tvorbou vývojového diagramu. Většinou si ale nebudete vytvářet vývojový diagram, pokud si budete něco zkoušet. Ale pokud se rozhodnete programovat nějaký program, který má být funkční, není nikdy na škodu vytvořit si vývojový diagram. Celkem dobře se v něm hledají chyby, můžete sledovat postup programu a případně vše poopravit. Vhodný program pro návrh či následnou úpravu vývojového diagramu je třeba Visio od společnosti Microsoft, o kterém jsem psal nedávno a nebo jak se říká – pokrok je pokrok, ale papír je papír. Zde je ovšem problém při editaci. Vývojový diagram není nutností, ale určitě bych ho doporučil obzvláště pokud pracujete jako skupina nebo pokud tvoříte rozsáhlejší projekt. Vývojový diagram se také hodí pokud přecházíte z jiného jazyka a chcete předělat nějaký program na jiný jazyk, tehdy je vývojový diagram velkým pomocníkem.

Vývojový diagram je tedy *grafické znázornění algoritmu*.

Algoritmus je přesný postup, který vede k vyřešení určitého výsledku. Vysvětlili jsme si pojem algoritmus a přidáme ještě jeden pojem a to **deterministický**. Znamená to, že pokud programu dáme určitá data, tak nám vrátí výsledek, a pokud mu tatáž data zadáme znovu, výsledek bude totožný s předchozím.

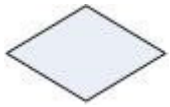
Vývojové diagramy se skládají z grafických značek. Značky jsou různé a různě se kombinují, tím se simulují různé situace a různé příkazy, do těchto značek se pak vypisují upřesňující údaje. Nyní se podíváme na to, jak vypadají jednotlivé části vývojového diagramu.

vyvdia1.1

Konec a začátek algoritmu

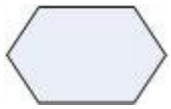
vyvdia1.2

Běžný příkaz



vyvdia1.3

Podmíněný výraz



vyvdia1.4

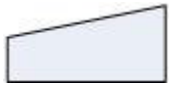
Cyklus s určeným počtem opakování

vyvdia1.5

Cyklus s podmínkou na konci

vyvdia1.6

Cyklus s podmínkou na začátku

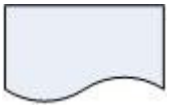


vyvdia1.7

Ruční vstup

vyvdia1.8

Zobrazení výstupu



vyvdia1.9

Zpracování souboru

vyvdia1.14

Uložení dat do souboru

vyvdia1.15

Podprogram

vyvdia1.12

Spojovací značka

vyvdia1.13

Spojovací čára

Kdybych začal od spojovací čáry, tak bych k ní podotkl, že pokud směřuje doprava nebo dolů, není k ní potřeba dělat šipku, pokud ovšem směřuje do leva či nahoru, tak se šipka dělat musí.

Při tvorbě vývojových diagramů se vychází z několika základních struktur a zvyklostí. My si je teď ukážeme a budeme se jich držet. Protože se standardně používají, tak ať si nevymýšlíme zbytečně vlastní...

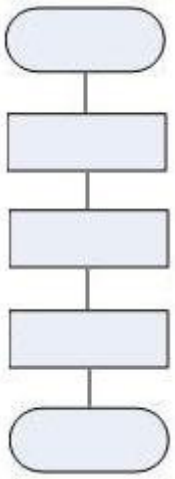
Sekvence:

Sekvence je první z možností. Jsou to příkazy jdoucí po sobě bez oklik, skoků a větvení, prostě řada příkazů, které se jeden po druhém provedou.

Např.:
- zajdi do obchodu
- udělej snídani
- uklid' si pokoj

Pokud bychom se chtěli více přiblížit výpočetní technice, mám tu jiný příklad.

- načti data do proměnných
- sečti proměnné
- vytiskni výsledek
- ukonči program



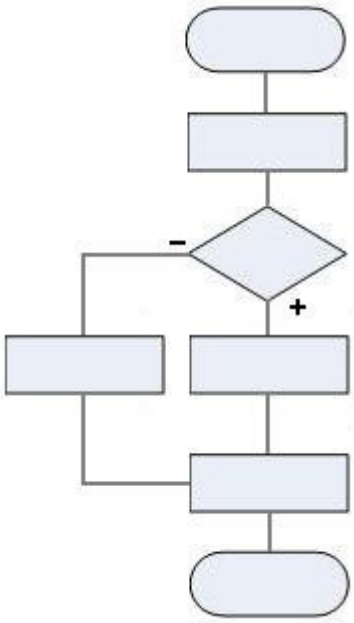
vyvDiag2.3

VĚTVENÍ:

Program se větví na několik částí. Která z těchto částí se vykoná, závisí často na podmínce. Pokud je tedy podmínka splněna, provede se něco jiného, než když podmínka splněna není.

Úplná alternativa.

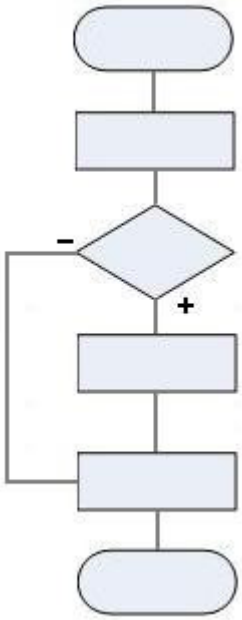
Program se větví do dvou částí, pokud je podmínka splněna, provede se jedna část kódu, pokud ne, provede se druhá část kódu. Např.: Pokud je součet zadaných čísel vyšší nebo roven nule, vytiskni, že je výsledek kladný, pokud je ale výsledek menší než nula, vytiskni, že je výsledek záporný.



vyvDiag2.1

Neúplná alternativa

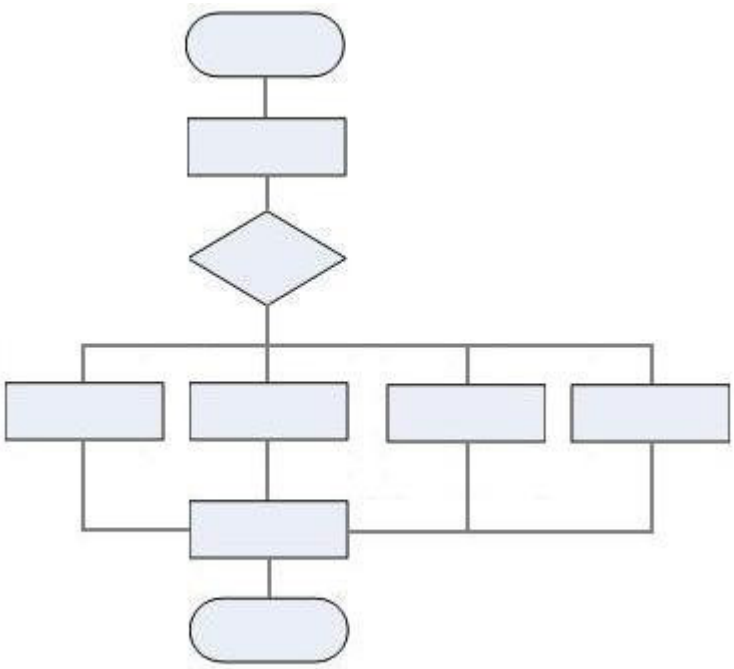
Jak je vidět na obrázku, neliší se tento případ příliš od předchozího, jen je jedna jeho větev prázdná. Znamená to, že pokud se podmínka splní, provede se zadaný kód a pak program pokračuje dalším příkazem. Pokud podmínka pravdivá není, pokračuje se za kódem podmínky v provádění dalšího kódu. Např.: Pokud proměnná x není prázdná, program ji vyprázdní, pokud je prázdná, program pokračuje dál.



vyvDiag2.2

Několikanásobná alternativa

Poslední způsob větvení, kterým se budeme zabývat, je několikanásobné větvení. Podle toho, čemu se rovná porovnávaná proměnná, se provede určitý kód. Např.: Dejme tomu, že po uživateli vyžadujeme, ať zadá nějaké písmeno a podle toho, jaké písmeno zadal, provedeme to, co uživatel chce. Právě v tomto případě se nám bude hodit tato několikanásobná alternativa.



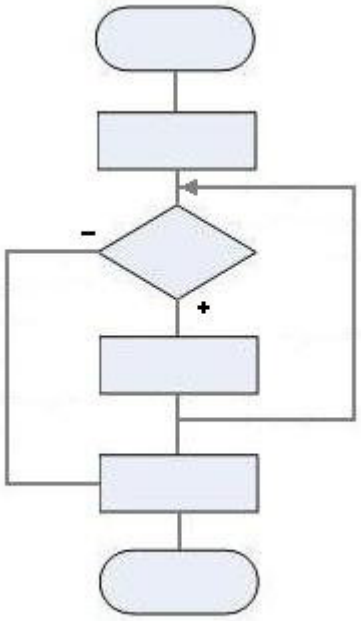
vyvDiag2.4

Tyto způsoby se dají různě kombinovat, vnořovat a proplétat a tak dál, ale teď se pustíme do cyklů.

CYKLY:

Se vstupní podmínkou

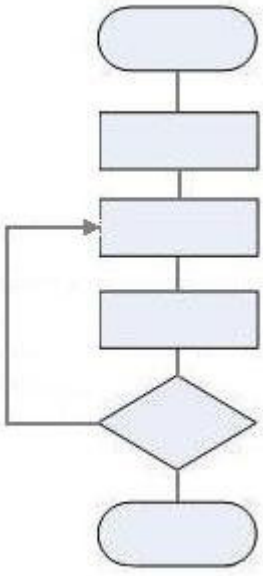
Tento cyklus se bude provádět, dokud si nebude výraz v podmínce cyklu roven. Pokud si výraz bude roven ještě předtím, než se začne cyklus provádět, cyklus se neprovede vůbec. Záleží také na typu cyklu. O těch příště.



vyvDiag2.5

S výstupní podmínkou

Tento cyklus je velmi podobný tomu předchozímu, ale podmínka je až na konci. Zaručuje nám to tedy, že tento cyklus se provede alespoň jednou.



vyvDiag2.6

Pokud byste neměli možnost využít cyklů a větveni, nebyly byste prakticky schopni tvořit plnohodnotné programy.