Programové struktury

- způsob zápisu algoritmu pomocí VD (vývojových diagramů)
- událostní funkce
- obecná funkce, objektová metoda, předávání parametrů funkci, návratová hodnota
- binární větvení
- cyklus (podmíněný, s definovaným počtem opakování)
- objekt
- příklady programů ve vybraném jazyce

Strukturované programování (strukturovaný programovací jazyk)

- Označuje v informatice **programovací techniku**, kdy se implementovaný algoritmus rozděluje na dílčí úlohy (tzn. na procedury či funkce), které se spojují v jeden celek. Na strukturované programování lze nahlížet jako na imperativní programování za využití vybraných řídicích struktur. U strukturovaného programování se vyhýbáme řídicímu příkazu skoku.

- Strukturované programování definuje, že se program může skládat pouze z následujících čtyř struktur:
  - **sekvence:** provádí posloupnost příkazů jeden po druhém
  - **větvení:** jeden nebo více příkazů je vykonán v závislosti na stavu programu (obvykle vyjadřováno klíčovými slovy if-else)
  - **cyklus:** příkazy jsou prováděny do té doby, dokud program nedosáhne nějakého stavu (obvykle vyjadřováno klíčovými slovy while, for)
  - **podprogram:** příkazy jsou shromažďovány do samostatného bloku, který má své jméno a definuje vstupy a výstupy. Tento blok (funkci nebo proceduru) lze z jiné části programu volat jeho jménem (identifikátorem). Funkce s návratovou hodnotou lze zařadit do výrazu.
- Nejznámějším důsledkem těchto zásad je snaha zabránit nebo v závislosti na programovacím jazyce alespoň omezit používání příkazu skoku. Programový kód nerespektující výše uvedené zásady se často hanlivě označuje jako „špagetový kód“.

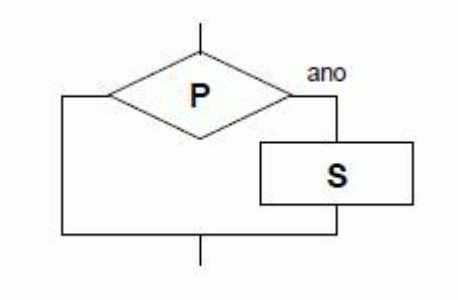
Sekvence

- Sekvence představuje posloupnost jednotlivých příkazů řazených v pořadí za sebou tak, jak mají být vykonány. Uzavřená sekvence se nazývá **složený příkaz nebo blok.**

Větvení

- Větvení umožňuje volit další postup řešení na základě splnění nebo nesplnění určité podmínky. Z obecného pohledu ale nemusí být vždy testována podmínka, neboť rozhodující pro větvení je až logická hodnota (true/false), která je výsledkem zmíněné podmínky. Větvit lze tedy i jen na základě hodnoty v logické proměnné. Větvení může být realizováno jako neúplný podmíněný příkaz, úplný podmíněný příkaz nebo jako přepínač.

1) Neúplný podmíněný příkaz

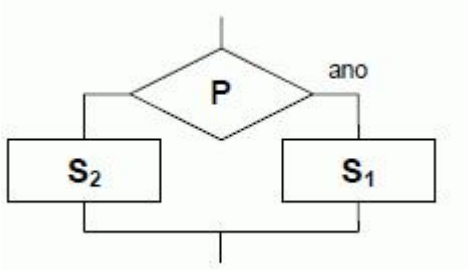


Nejprve je vyhodnocena podmínka P. Když je splněna, provede se sekvence S. Není-li podmínka splněna, nic se neprovádí a běh programu pokračuje dále.

C#	VB

`int x;`
`if (x > 0) MessageBox.Show("Kladné číslo");`
	`Dim x As Integer`
	`If (x > 0) Then`
	`    MsgBox "Kladné číslo"`
	`End If`

2) Úplný podmíněný příkaz



Nejprve je vyhodnocena podmínka P. Když je splněna, provede se sekvence S1. Není-li podmínka splněna, provede se sekvence S2.

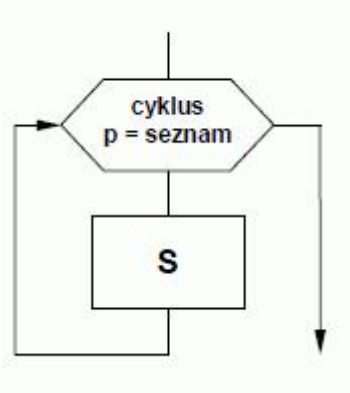
C#	VB

`int x;`
`if (x > 0)`
`{`
`        MessageBox.Show("Je přirozené číslo");`
`}`
`else`
`{`
`        MessageBox.Show("Není přirozené číslo");`
`}`
	`Dim x As Integer`
	`If (x > 0) Then`
	`        MsgBox "Je přirozené číslo"`
	`Else`
	`        MsgBox "Není přirozené číslo"`
	`End If`

Cyklus

- Slouží k vyjádření opakujících se operací. Počet opakování může být předem zadán, nebo je odvozen od splnění některé dané podmínky. Existují tak tři základní druhy cyklu: **cyklus s výčtem hodnot, cyklus s podmínkou na počátku a cyklus s podmínkou na konci.**

1) Cyklus s výčtem hodnot (cyklus s předepsaným počtem opakování)

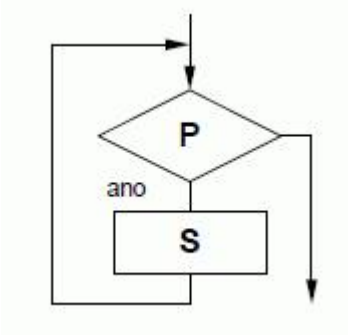


Tento druh cyklu se provádí tolikrát, kolikrát je opakování cyklu předepsáno. Předpis je dán definicí dolní hodnoty p, kroku a horní hodnoty p (případně podmínky, která musí být po celou dobu vykonávání cyklu splněna).

C#	VB

`for (int i = 0; i < 10; i++) x[i] = nahoda.next(0, 11);`	`Dim i As Integer`
	`Dim x(10) As Double`
	`Randomize`
	`For i = 0 To 9`
	`  x(i) = 11 * Rnd`
	`Next i`

2) Cyklus s podmínkou na začátku



Nejprve je vyhodnocena podmínka P. Je-li podmínka splněna, provedou se příkazy sekvencí C a následuje návrat na začátek cyklu. Tato se opakuje, pokud stále platí podmínka P. Z uvedeného je zřejmé, že v sekvenci S musí být operace, která mění některou z hodnot logického výrazu podmínky P. Kdyby tomu tak nebylo, cyklus by nemohl skončit. Při dosažení neplatnosti podmínky P cyklus skončí.

Je tedy zřejmé, že cyklus s podmínkou na začátku (někdy zvaný cyklus POKUD) nemusí proběhnout vůbec (je-li podmínka P nesplněna před vstupem do cyklu), nebo jednou, či n-krát. V případě chybného naprogramování cyklu může dojít k zacyklení, což nastane, pokud podmínka P platí stále.

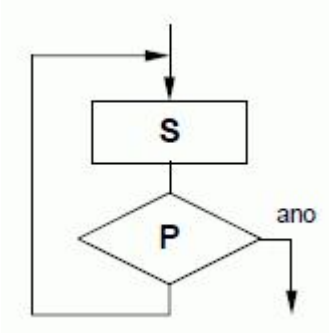
C#	VB

```
int i=0;
while (i<10)
{
        x[i] = nahoda.next(0, 11);
        i++;
}
```

```
Dim i As Integer
Dim x(9) As Double
i = 0
Randomize

Do While (i < 10)
        x(i) = 11 * Rnd
        i = i + 1
Loop
```

3) Cyklus s podmínkou na konci



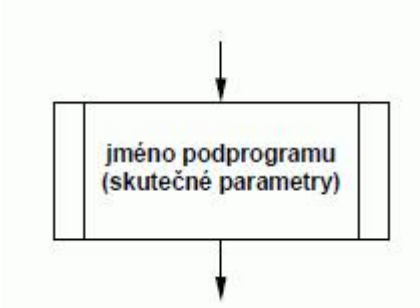
Nejprve je provedena sekvence S. Poté je vyhodnocena podmínka, pokud platí, sekvence S se znovu neprovádí a běh programu pokračuje za cyklem. Není-li podmínka splněna opakuje se sekvence S a to tak dlouho, dokud podmínka stále neplatí. Z uvedeného je zřejmé, že v sekvenci S musí být operace, která mění některou z hodnot logického výrazu podmínky P. Kdyby tomu tak nebylo, cyklus by nemohl skončit.

Tento cyklus se někdy nazývá cyklus DOKUD, čímž se myslí - dokud podmínka neplatí, opakuje se sekvence S. Tento cyklus proběhne vždy alespoň jednou. Četnost výskytu potřeby tohoto typu cyklu v programech je asi desetkrát menší než cyklu s podmínkou na začátku.

C#	VB

`int i=0;`
`do`
`{`
`        x[i] = nahoda.next(0, 11);`
`        i++;`
`} while (i<10)`
	`Dim i As Integer`
	`Dim x(9) As Double`
	`i = 0`
	`Randomize`
	
	`Do`
	`        x(i) = 11 * Rnd`
	`        i = i + 1`
	`Loop While (i < 10)`

Podprogram



- Podprogramy představují ucelené, relativně samostatné části programu, napsané v tomto programu jenom jednou, ale s možností využít je vícekrát v různých místech tohoto programu. Podprogram musí mít své jméno (identifikátor), jehož prostřednictvím se na něj v programu odvoláváme a seznam formálních parametrů, které slouží po jejich nahrazení skutečnými parametry při odvolání se na podprogram.

- Podprogram se může vyskytovat v podobě funkce bez návratové hodnoty nebo s návratovou hodnotou. Funkci s návratovou hodnotou můžeme zahrnout do výrazu.
  - Funkce, která je volána při nějaké události (kliknutí na tlačítko, stisknutí klávesy,...), se nazývá jako **událostní funkce** (také událostní procedura).

Parametry předávané do podprogramu

- Parametr je označení pro vstupní data příslušné funkce. Existují dva způsoby předávání parametrů funkci.
  - Při **předávání hodnotou** (call-by-value) se těsně před zpracováním těla funkce předávaný parametr vyčíslí a výsledek se zkopíruje do lokální proměnné uvnitř volané funkce. Jakékoli změny parametru uvnitř volané funkce nemají vliv na volající funkci, neboť se pracuje s lokální kopií hodnoty původního parametru. Předávání hodnotou tedy lze používat pouze pro vstupní parametry. Tento způsob je typický např. při vytváření aritmetických funkcí.
  - Při **předávání odkazem** (call-by-reference) se formální parametr uvnitř volané funkce bere jen jako jiné označení (alias) pro proměnnou předanou jako skutečný parametr, tzn. ve volané funkci se pracuje přímo s předávanou proměnnou, nevytváří se tedy kopie hodnoty (což zvláště u strukturovaných proměnných znamená zpravidla úsporu času i paměti). Volaná funkce změnou parametru ovlivňuje i původní volající funkci, takže předávání odkazem lze používat pro výstupní či vstupně-výstupní parametry. Nevýhodou však je, že parametrem může být jen proměnná a nikoli výsledek obecného výrazu. Předávání odkazem se obvykle implementuje pomocí ukazatele na předávanou proměnnou.

1) Předávání parametrů hodnotou

C#	VB

```
void Zamen(int a, int b) //zameni
hodnoty v promennych
{
        int pom;
        pom = a;
        a = b;
        b = pom;
}
```

Tuto funkci budeme volat z volající funkce následně:

```
static void Main(string[] args)
{
        int promennaA = 10, promennaB
= 20;
        Zamen(promennaA, promennaB);
        return 0;
}
```

```
Private Sub Zamen(ByVal a As Integer,
ByVal b As Integer)
        Dim pom As Integer
        pom = a
        a = b
        b = pom
End Sub
```

Tuto proceduru (funkci) budeme volat z následující funkce následně:

```
Private Sub NejakaFunkce()
        Dim promennaA As Integer
        Dim promennaB As Integer
        promennaA = 10
        promennaB = 20
        Zamen promennaA, promennaB
End Sub
```

Při předávání parametrů funkci hodnotou zůstanou původní proměnné (`promennaA` a `promennaB`) po návratu zpět do volající funkce beze změny.

2) Předávání parametrů odkazem

C#

```
void Zamen(ref int a, ref int b)
//zameni hodnoty v promennych
{
        int pom = a;
        a = b;
        b = pom;
}
```

Tuto funkci budeme volat z volající funkce následně:

```
static void Main(string[] args)
{
        int promennaA = 10, promennaB
= 20;
        Zamen(ref promennaA, ref
promennaB);
        return 0;
}
```

VB

```
Private Sub Zamen(ByRef a As Integer,
ByRef b As Integer)
        Dim pom As Integer
        pom = a
        a = b
        b = pom
End Sub
```

Tuto proceduru (funkci) budeme volat z následující funkce následně:

```
Private Sub NejakaFunkce()
        Dim promennaA As Integer
        Dim promennaB As Integer
        promennaA = 10
        promennaB = 20
        Zamen promennaA, promennaB
End Sub
```

Po předání parametrů funkci tímto způsobem budou po návratu zpět do volající funkce v našich proměnných (`promennaA` a `promennaB`) hodnoty 20 a 10.

Objekt

*Poznámka: Pojem **objekt** souvisí s objektovým programování. Podle Procházkových slov je objektové programování vyučováno až na vysoké škole. U maturity tento pojem pravděpobně nezazní. Pokud přece jen ano, můžete říci, že jste se s objekty setkali především při programování v **C#** (ostatně i ve VB - za objekt se tam dá považovat každé tlačítko, vstupní pole, zaškrtávátko, ...). Objekt je datová struktura, která seskupuje určité **vlastnosti** (tj. proměnné objektu) a **metody** (funkce, které pracují s vlastnostmi objektu).*

Objektem může být cokoliv - např. člověk. Vlastnostmi (proměnnými) by poté bylo např. jméno, věk, datum narození, počet životů, barva očí, apod. Metodami (funkcemi) např. změnBarvuVlasů(), jdiSpát(), najezSe(), jdiNaMísto(xyz), pracuj(), apod.