1. Úvod do programování

B0B99PRPA – Procedurální programování
A8B14ADP – Algoritmizace a programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Čím se budeme v předmětu zabývat

- Řešení problémů pomocí počítačových programů
- Programováním v jazyce C
- Fungování počítačů
- Jednoduchými, ale důležitými algoritmy
- Složitostí algoritmů

Znalosti

Deklarativní znalost je tvrzení či výrok, u nějž lze určit, zda je pravdivý.

$\sqrt{x}$ je y, pro které platí $y^2 = x$.

Imperativní znalost je receptem, jak něčeho dosáhnout.

Nechť g je odhadem y

- pokud je $g^2 \approx x$, ukonči program a vrať g
- nový odhad je dán $\frac{g+\frac{x}{g}}{2}$
- opakuj, dokud nejsi spokojen s hodnotou g

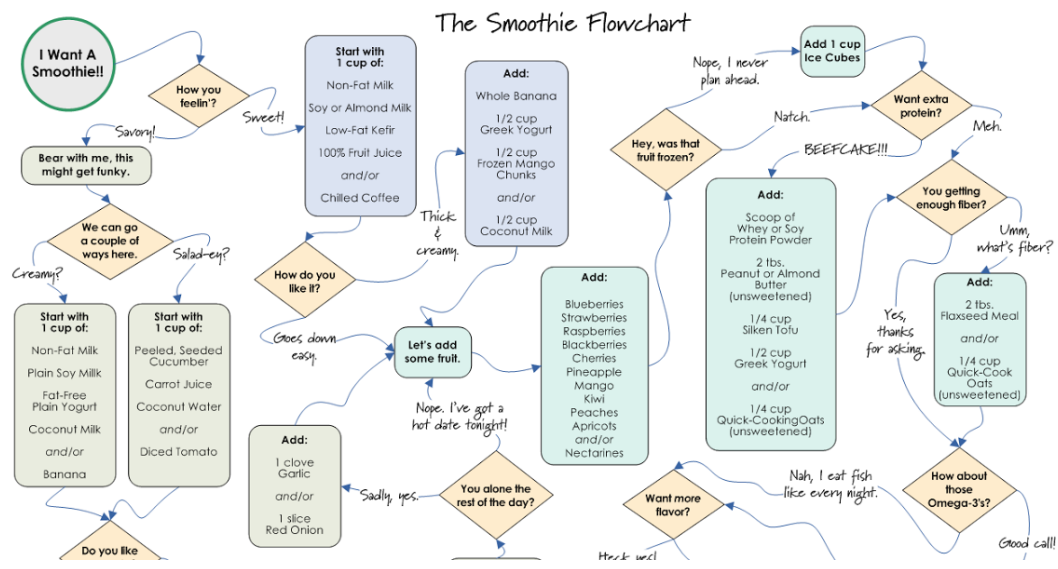
Algoritmus

- Návod nebo postup, jak provést určitou činnost.
- Algoritmus by měl být tak podrobný, aby mu porozumněl i počítač.
- Vlastnosti algoritmu:
  1. Skládá se z konečného počtu jednoduchých činností – kroků.
  2. Po každém kroku lze určit, jak se má pokračovat nebo skončit.
  3. Počet opakování jednotlivých kroků algoritmu je vždy konečný.
  4. Vede ke správnému výsledku.
  5. Algoritmus lze použít k řešení celé (velké) skupiny podobných úloh.

Manželka: kup chleba a když budou mít rohlíky, vezmi jich deset.
Manžel, programátor: přinese z obchodu deset chlebů, protože rohlíky měli.

Slovo algoritmus vzniklo odvozením od jména perského matematika Al-Chorezmího, jehož jméno bylo ve středověku latinizováno jako Al-Gorizmí.

Algoritmus je recept



Základní části algoritmů

- Algoritmus zpravidla transformuje množinu vstupních dat na množinu dat výstupních
- Základní části algoritmů

  Vstup dat – načtení dat programem, interaktivní nebo ze souboru dat
  Popis dat – volba datového typu a umístění v paměti
  Zpracování – výpočet definovaný algoritmem, řízení toku programu
  Výstup – interakce s uživatelem nebo např. zápis do souboru
- Řízení toku algoritmu

  Posloupnost – jeden nebo několik kroků, které se provedou právě jednou v daném pořadí
  Cyklus – opakování nějaké posloupnosti, dokud je splněna podmínka opakování
  Větvení – volba posloupnosti instrukcí na základě vyhodnocení podmínky
- Kombinace základních složek algoritmu umožňuje vytvářet komplexní konstrukce.
- Pokud se některé části algoritmu opakují, je vhodné poslupnosti orgranizovat do větších celků: procedur a funkcí (podprogramů).

Počítače

- Počítače vykonávají algoritmy
- Umí dvě hlavní věci
  - provádět jednoduché **operace**
  - **pamatovat** si výsledky
- K vykonávání programů používají vlastní instrukce
  - Alan Turing dokázal, že libovolný program lze realizovat pomocí 6 základních instrukcí
- Programovací jazyky vytváří abstraktní vrstvu pro programátora

Jak začít?

Jednoduché algoritmy, grafické programování

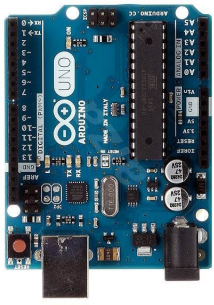
- Scratch – MIT Media Lab
- Angry Birds
- Code with Anna and Elsa
- Minecraft

Programovací jazyk Karel

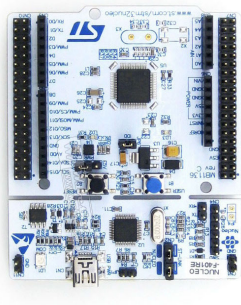
- Pohyb robota po čtvercové sítí
- Richard E. Pattis, Karel The Robot: A Gentle Introduction to the Art of Programming, Stanford, 1981
- Online: Stanford , Oldřich Jedlička

Další zajímavé programovací jazyky pro výuku programování např. zde .

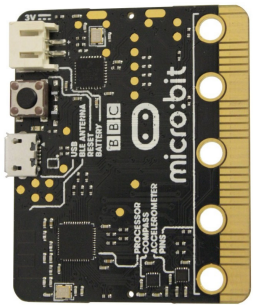
Programování může být skvělá zábava



Arduino
Open Source
Procesory AVR



Nucleo
ST Microelectronics
Procesory ARM



BBC Micro:bit
Open Source
Procesory ARM

Programovací jazyk

První program

Druhý program

Třetí program

Struktura zdrojového kódu

Programovací jazyk? C

... 1960s

Bell Labs

Ken Thompson

Space Travel

GE 635

PDP-7

Unix

Dennis Ritchie

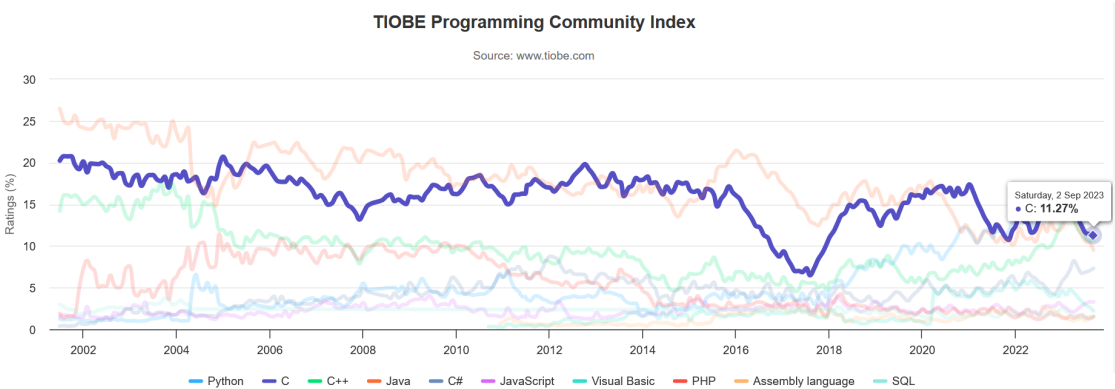
B → C

Brian Kernigan

K&R

ANSI C (C89)

Proč C?


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
Saturday, 2 Sep 2023
C: 11.27%
Python
C
C++
Java
C#
JavaScript
Visual Basic
PHP
Assembly language
SQL

Proč C? 2/2

Quora

- Is it necessary to learn C in 2019?
- Is learning C still worthwhile?
- Why does the C programming language refuse to die?
- Can I learn C language in one month?

Benchmark Game

- Which programs are fastest?
- Práce s poli: Read DNA sequences - write their reverse-complement
- Matematické operace: Eigenvalue using the power method

Srovnávat rychlost jazyků není snadné – záleží na úloze, datech, CPU, cache, ...

Jaký je C jazyk? 1/2

- Nízkoúrovňový – přístup k nízkoúrovňovým funkcím počítače

  Operace souvisí s operacemi definovanými platformou.

- Malý

  Malá množina výrazových prostředků + knihovna standardních funkcí.

- Procedurální (ne objektový)

  Programy se skládají pouze z funkcí a dat.

- Kompilovaný – překlad do strojového kódu

  Interpretované – potřebují běhové prostředí

- Staticky typovaný – kontrola typu proměnné při kompilaci

  Dynamicky typované jazyky – kontrola typu za běhu.

- Imperativní – chování programu je popsáno algoritmem

  Jiným typem je behaviorální – popis vstupů a výstupů.

Jaký je C jazyk? 2/2

Vhodný pro

- rychlé vědecké výpočty
- systémové aplikace
- programování s přístupem na hardware (vestavná zařízení, IoT, ...)
- rychlá grafika, hry, ...

Nevhodný pro

- webové aplikace (vhodnější jsou např. JavaScript, PHP, ...)
- rychlé prototypy (vhodnější např. Python)
- větší projekty, objektový přístup (vhodnější C++, Java, ...)

Zajímavé projekty/programy v C

- Linux , Mars Curiosity Rover , Bloomberg's distributed RDBMS , Python , GIMP

Jazyk C a další

- C/C++
  - překlad přímo do strojového kódu
  - překlad nutný pro každou platformu (procesor) zvlášť
- Další imperativní: Java, C#, ...
  - překlad do bytecode, jeden binární soubor pro více platforem
  - JVM (Java Virtual Machine) pro velké množství platforem
  - bytecode je interpretovaný, ale JIT (Just in Time) kompilátor
- Skriptovací imperativní: Python, Perl, ...
  - typicky se interpretuje, platformově nezávislé (pokud je interpret)
- Funcionální: Haskel, LISP...
  - jiné paradigma: matematický zápis odvození z počátečních hodnot
- Logické programování: Prolog, ...
  - jiné paradigma: JAK má výsledek vypadat, ne jak se k němu dostat

V zásadě by sem bylo možné zařadit i jazyk SQL

Zápis a překlad programu

- **Zdrojový kód**
  - běžný textový soubor
  - zdrojové soubory zpravidla s příponou **.c**
  - hlavičkové soubory s koncovkou **.h**
- **Kompilace**
  - kompilací zdrojového kódu vzniká binární objektový kód
  - z objektových souborů se sestavuje spustitelný kód
- **Vývoj v C**
  - Textový editor – gedit, atom, sublime, vim
  - Překladače gcc a clang
  - Sestavení projektu nástrojem make
  - Všechny kroky bývají integrovány v C/C++ vývojových prostředích
    - Visual Studio Code, Geany, Code::Blocks, NetBeans, Eclipse

Programovací jazyk

První program

Druhý program

Třetí program

Struktura zdrojového kódu

P1.1 Posloupnost

```
1  #include <stdio.h>
3  int main()
4  {
5    printf("Hello ");
6    printf("PRPA!\n");
7    return 0;
8  }
```

Někde na disku existuje soubor *stdio.h*, který potřebuji k překladu.

Kód spustitelného programu obsahuje funkci main().

Kód je organizován do bloků ohraničených {}.

Funkce printf tiskne text na displej.

Program (funkce) má návratovou hodnotu.

- Program můžeme zkompilovat a spustit

```
$ gcc hello.c
```

- Na displej počítače (standardní výstup) se vypíše textová informace

```
$ ./a.out
Hello PRPA!
```

P1.1 Posloupnost

```
1  #include <stdio.h>
3  int main()
4  {
5    printf("Hello ");
6    printf("PRPA!\n");
7    return 0;
8  }
```

Někde na disku existuje soubor *stdio.h*, který potřebuji k překladu.

Kód spustitelného programu obsahuje funkci main().

Kód je organizován do bloků ohraničených {}.

Funkce printf tiskne text na displej.

Program (funkce) má návratovou hodnotu.

- Program můžeme zkompilovat a spustit

```
$ gcc hello.c
```

- Na displej počítače (standardní výstup) se vypíše textová informace

```
$ ./a.out
Hello PRPA!
```

P1.1 Posloupnost

```
1  #include <stdio.h>
3  int main()
4  {
5    printf("Hello ");
6    printf("PRPA!\n");
7    return 0;
8  }
```

Někde na disku existuje soubor *stdio.h*, který potřebuji k překladu.

Kód spustitelného programu obsahuje funkci main().

Kód je organizován do bloků ohraničených {}.

Funkce printf tiskne text na displej.

Program (funkce) má návratovou hodnotu.

- Program můžeme zkompilovat a spustit

```
$ gcc hello.c
```

- Na displej počítače (standardní výstup) se vypíše textová informace

```
$ ./a.out
Hello PRPA!
```

P1.1 Posloupnost

```
1  #include <stdio.h>
3  int main()
4  {
5    printf("Hello ");
6    printf("PRPA!\n");
7    return 0;
8  }
```

Někde na disku existuje soubor *stdio.h*, který potřebuji k překladu.

Kód spustitelného programu obsahuje funkci main().

Kód je organizován do bloků ohraničených {}.

Funkce printf tiskne text na displej.

Program (funkce) má návratovou hodnotu.

- Program můžeme zkompilovat a spustit

```
$ gcc hello.c
```

- Na displej počítače (standardní výstup) se vypíše textová informace

```
$ ./a.out
Hello PRPA!
```

P1.1 Posloupnost

```
1  #include <stdio.h>
3  int main()
4  {
5    printf("Hello ");
6    printf("PRPA!\n");
7    return 0;
8  }
```

Někde na disku existuje soubor *stdio.h*, který potřebuji k překladu.

Kód spustitelného programu obsahuje funkci main().

Kód je organizován do bloků ohraničených {}.

Funkce printf tiskne text na displej.

Program (funkce) má návratovou hodnotu.

- Program můžeme zkompilovat a spustit

```
$ gcc hello.c
```

- Na displej počítače (standardní výstup) se vypíše textová informace

```
$ ./a.out
Hello PRPA!
```

Programovací jazyk

První program

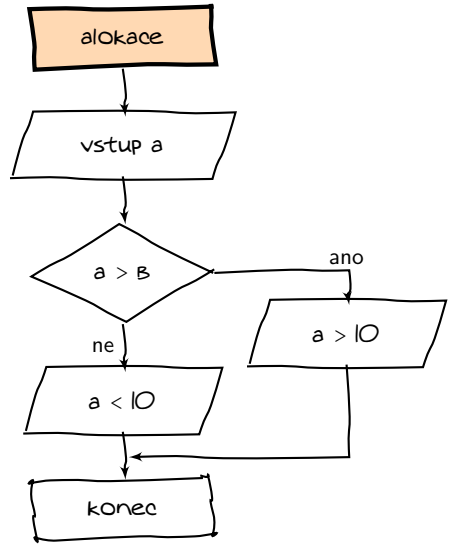
Druhý program

Třetí program

Struktura zdrojového kódu

P1.2 Větvení

```
1  #include <stdio.h>
3  int main()
4  {
5    int a;
7    scanf ("%d", &a);
9    if (a > 10)
10     printf ("a > 10");
11   else
12     printf ("a <= 10");
14   return 0;
15 }
```



- Program načte číslo a oznámí výsledek

```
$ echo 20 | ./a.out
a > 10
```

P1.2 Větvení

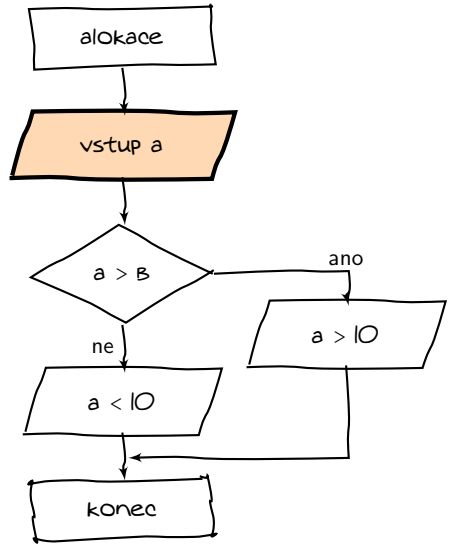
```c
#include <stdio.h>

int main()
{
  int a;

  scanf ("%d", &a);

  if (a > 10)
    printf ("a > 10");
  else
    printf ("a <= 10");

  return 0;
}
```



- Program načte číslo a oznámí výsledek

```
$ echo 20 | ./a.out
a > 10
```

P1.2 Větvení

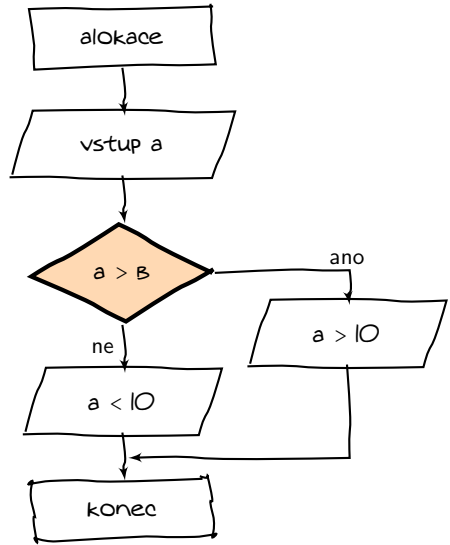
```c
#include <stdio.h>

int main()
{
  int a;

  scanf ("%d", &a);

  if (a > 10)
    printf ("a > 10");
  else
    printf ("a <= 10");

  return 0;
}
```

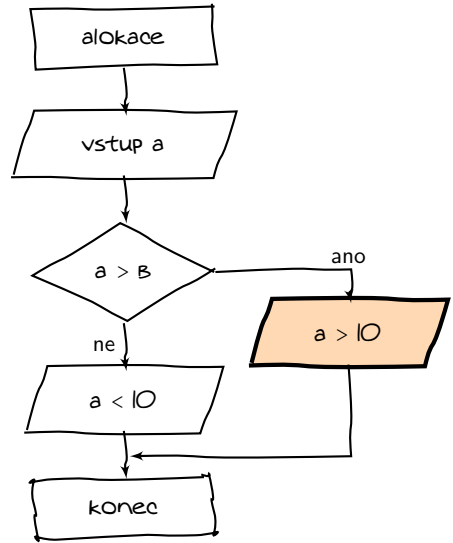


- Program načte číslo a oznámí výsledek

```
$ echo 20 | ./a.out
a > 10
```

P1.2 Větvení

```
1  #include <stdio.h>
3  int main()
4  {
5    int a;
7    scanf ("%d", &a);
9    if (a > 10)
10     printf ("a > 10");
11   else
12     printf ("a <= 10");
14   return 0;
15 }
```



- Program načte číslo a oznámí výsledek

```
$ echo 20 | ./a.out
a > 10
```

P1.2 Větvení

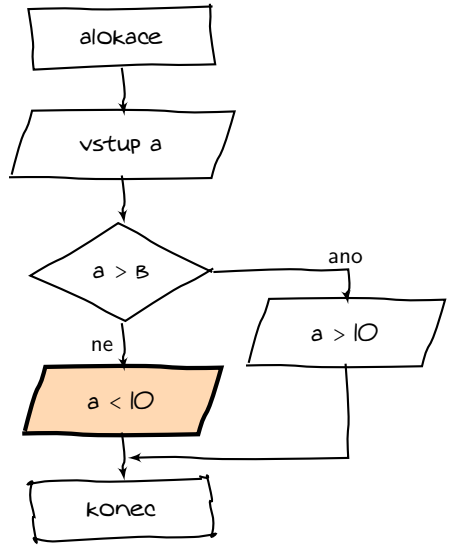
```c
#include <stdio.h>

int main()
{
  int a;

  scanf ("%d", &a);

  if (a > 10)
    printf ("a > 10");
  else
    printf ("a <= 10");

  return 0;
}
```



- Program načte číslo a oznámí výsledek

```
$ echo 20 | ./a.out
a > 10
```

P1.2 Větvení

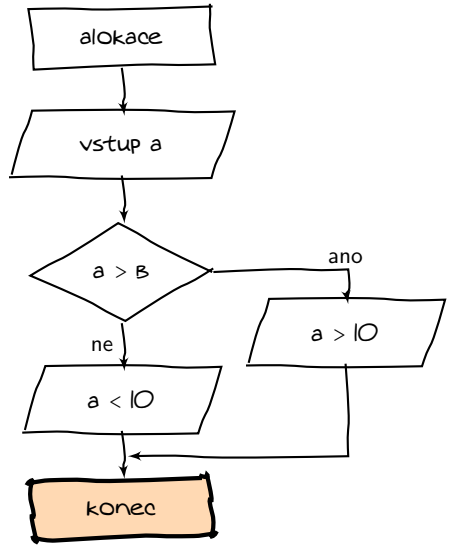
```c
#include <stdio.h>

int main()
{
  int a;

  scanf ("%d", &a);

  if (a > 10)
    printf ("a > 10");
  else
    printf ("a <= 10");

  return 0;
}
```



- Program načte číslo a oznámí výsledek

```
$ echo 20 | ./a.out
a > 10
```

Programovací jazyk

První program

Druhý program

Třetí program

Struktura zdrojového kódu

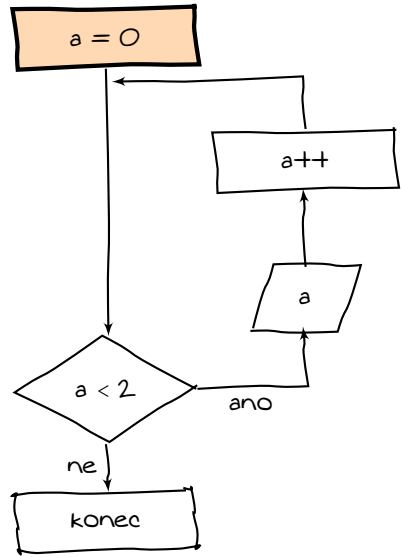
P1.3 Cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;
  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }
  return 0;
}
```

- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```

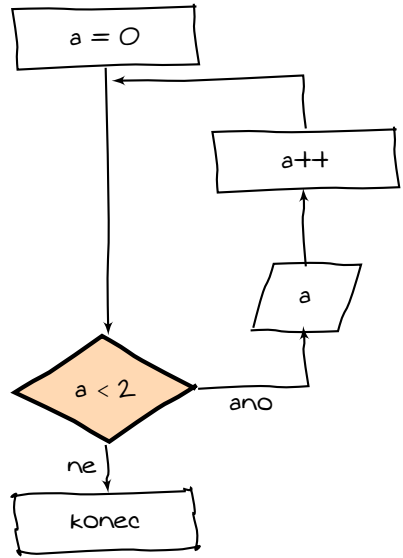


P1.3 Cyklus

```
1  #include <stdio.h>
3  int main()
4  {
5    int a = 0;
7    while (a < 2)
8    {
9      printf ("%d", a);
10     a++;
11   }
13   return 0;
14 }
```

- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```



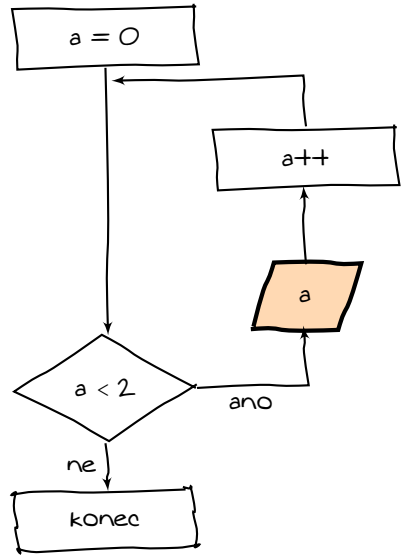
P1.3 Cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;
  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }
  return 0;
}
```

- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```



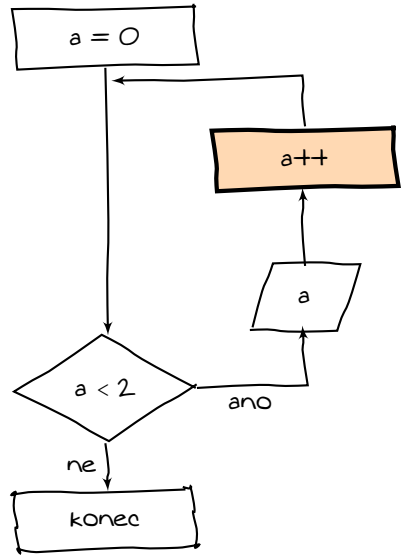
P1.3 Cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;
  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }
  return 0;
}
```

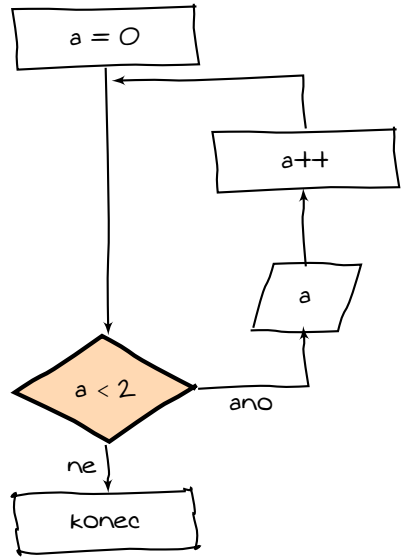
- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```



P1.3 Cyklus

```
1  #include <stdio.h>
3  int main()
4  {
5    int a = 0;
7    while (a < 2)
8    {
9      printf ("%d", a);
10     a++;
11   }
13   return 0;
14 }
```



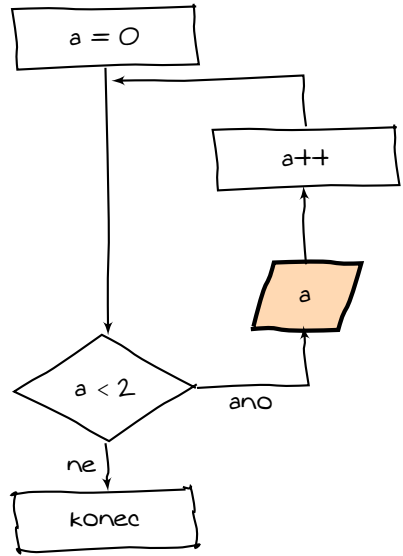
- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```

P1.3 Cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;
  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }
  return 0;
}
```



- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```

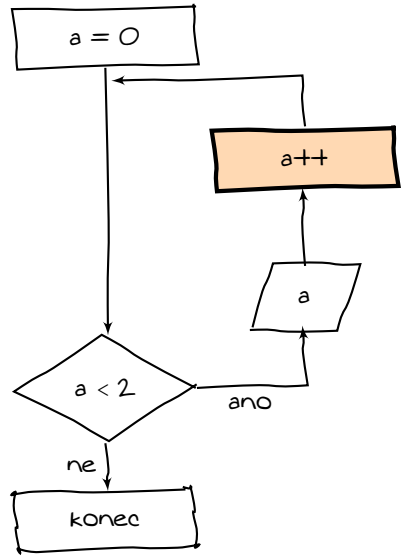
P1.3 Cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;
  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }
  return 0;
}
```

- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```

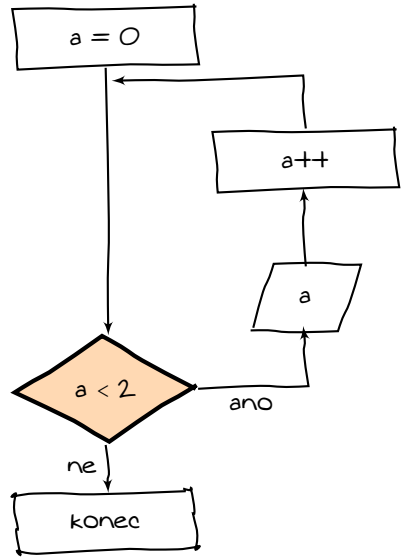


P1.3 Cyklus

```
1  #include <stdio.h>
3  int main()
4  {
5    int a = 0;
7    while (a < 2)
8    {
9      printf ("%d", a);
10     a++;
11   }
13   return 0;
14 }
```

- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```



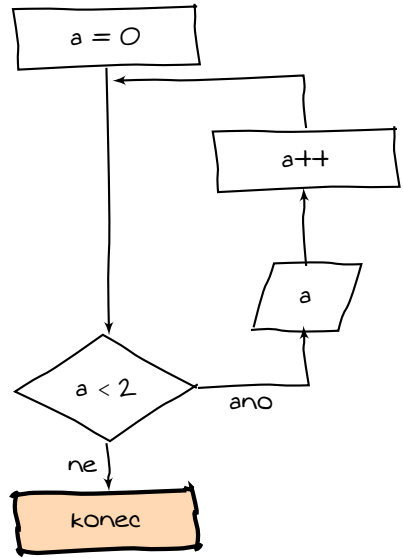
P1.3 Cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;
  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }
  return 0;
}
```

- Na displej se vypíše řada čísel

```
$ ./a.out
0 1
```



Programovací jazyk

První program

Druhý program

Třetí program

Struktura zdrojového kódu

Struktura zdrojového kódu

```c
/* vlozeni hlavickoveho souboru
   standardni knihovny */
#include <stdio.h>
// hlavicka funkce main()
int main()
{ // hlavni funkce programu
  /*
     volani funkce definovane
     v knihovne stdio.h
  */
  printf("Uz programuju!\n");
  /*
     ukonceni programu a predani
     navratove hodnoty 0
     operacnimu systemu
  */
  return 0;
}
```

Direktivy kompilátoru

- Příkazy uvozené znakem `#`

Klíčová slova jazyka C

- Zde `int` a `return`

Identifikátory

- Jména proměnných, funkcí a typů
- Alfanumerické znaky a podtržítko, case sensitive

Komentáře

- Text umístěný mezi dvojici párových symbolů `/*` a `*/`
- Je možné i použití `//`

Odsazovače

- Mezera, tabelátor, nový řádek, ...

Závorky

- Párové symboly `{}`, `()`, `<>`

Klíčová slova

Žádný identifikátor nemůže mít v době překladu stejný název

C89

`auto`	`break`	`case`	`char`	`const`	`continue`	`default`	`do`
`double`	`else`	`enum`	`extern`	`float`	`for`	`goto`	`if`
`int`	`long`	`register`	`return`	`short`	`signed`	`sizeof`	`static`
`struct`	`switch`	`typedef`	`union`	`unsigned`	`void`	`volatile`	`while`

C99

`inline`	`restrict`	`_Bool`	`_Complex`	`_Imaginary`