1. Informace o předmětu, úvod do programování

B0B99PRPA – Procedurální programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Přehled témat

Část I

O předmětu

I. O předmětu

Předmět a lidé

- Webové stránky předmětu

  https://cw.fel.cvut.cz/wiki/courses/b0b99prpa/start

- Přednášející a garant předmětu
  - Stanislav Vítek, viteks@fel.cvut.cz

    http://mmtg.fel.cvut.cz/personal/vitek/
- Cvičící
  - Martin Mudroch, mudromar@fel.cvut.cz
  - Ondřej Nentvich, nentvond@fel.cvut.cz
  - Václav Navrátil, vaclav.navratil@fel.cvut.cz
  - Václav Vencovský, vecovac@fel.cvut.cz
- Konzultace
  - distanční výuka – MS Teams, individuálně / skupinově
  - kontaktní výuka – pátek 9:00-10:30, H131

Cíle předmětu

- **Motivovat k programování**
  - Programování je klíčová dovednost, která může hrát rozhodující roli na trhu práce
- **Naučit se algoritmizovat**
  - Formulace problému a návrh řešení
  - Rozklad problému na dílčí úlohy
  - Identifikace opakujících se vzorů
- **Získat zkušenosti s programováním**
  - Základní programovací konstrukce

    Proměnné, cykly, podmínky, datové struktury a jednodušší algoritmy
  - Programovací jazyk C, řada principů obecně použitelných

    Cvičení, domácí úkoly, hledání chyb, práce s dokumentací, test

Programátorovi nestačí perfektní znalost programovacího jazyka, ale především musí vědět, jak vůbec danou úlohu řešit.

Organizace a hodnocení předmětu

- **Studijní výsledky**
  - Průběžná práce v semestru – domácí úkoly a test
  - Zápočtový a případně implementační test
- **Docházka**
  - Přednášky jsou nepovinné, ale snad přínosné a zábavné
  - Cvičení jsou povinná, možné dvě omluvené absence
  - Na cvičení se očekává aktivní účast při řešení příkladů

Na cvičení je třeba se **připravit**, nejlépe návštěvou přednášky a studiem podkladů (řešené příklady)

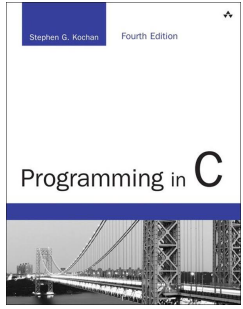
- **Řešení problémů**
  - Obracejte se na svého cvičícího
  - Při komunikaci e-mailem pište vždy ze své fakultní adresy
  - Do předmětu zprávy uvádějte zkratku předmětu PRPA
  - V případě zásadních problémů uvádějte do CC též přednášejícího

Zdroje a literatura

Pavel Herout
Učebnice jazyka C

Kopp, 2011

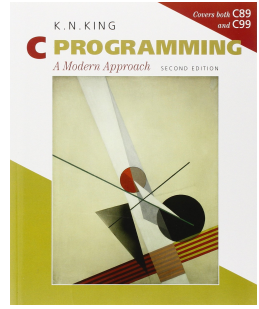
ISBN 978-80-7232-383-8



Stephen G. Kochan
Programming in C

Addison-Wesley 2014

ISBN 978-0321776419

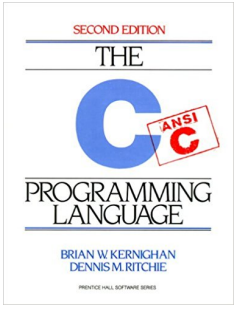


K. N. King
C Programming: A Modern Approach

W. W. Norton & Company 2008

ISBN 860-1406428577

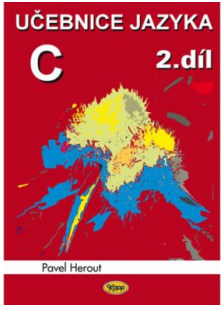
Zdroje a literatura



Brian W. Kernighan
Dennis M. Ritchie
The C Programming Language (ANSI C)

Prentice Hall, 1988

ISBN 978-0131103627



Pavel Herout
Učebnice jazyka C – 2. díl

Kopp, 2008

ISBN 978-80-7232-367-8

Peter van der Linden
Expert C Programming: Deep C Secrets

Prentice Hall, 1994

ISBN 978-0131774292

I. O předmětu

Domácí úkoly

- Samostatná práce, praktické zkušenosti s programováním
- Jednotné zadání na prednášce a jednotný termín odevzdání
- Náročnost domácích úkolů se postupně zvyšuje
- Odevzdání domácích úkolů v systému BRUTE
- Cílem řešení úkolů je získat **vlastní** zkušenost
  - Neopisujte – škodíte především sobě
  - Automatická kontrola plagiátů u všech odevzdaných řešení
    - každý s každým
    - každý s řešením z minulých let (pokud je podobný příklad)
    - u podezdřelých případů provedeme manuální kontrolu
  - V případě odhalení jsou potrestání **oba** účastníci incidentu

Pokud nečemu nerozumíte, ptejte se!

Přehled domácích úkolů

HW01 – První program, Hello PRPA!	(**1**)
HW02 – Načítání vstupu, výpočet a výstup	(**2**)
HW03 – První cyklus	(**3**)
HW04 – RLE kodér	(**4**)
HW05 – Kreslení (ASCII art)	(**2**+2+3)
HW06 – Caesarova šifra	(**4**)
HW07 – Maticové počty	(**2**+2+3)
HW08 – Zpracování číselné řady	(**4**)
HW09 – Analýza textového souboru	(**4**+2+2)
HW10 – Kruhová fronta v poli	(**4**)
HW11 – Zpracování strukturovaného textu	(**0**+4+3)

Celkem lze získat **30b** za povinná zadání a dalších **20b** za bonusová.

Kontrola domácích úkolů – odevzdávací systém BRUTE

- Formální kontrola – kompilace programu
- Testování funkčnosti a správnosti – kontrola výstupu pro daný vstup
  - Veřejné vstupy a odpovídající výstupy / neveřejné vstupy
- Před uploadem programu si program otestujete sami
  - S využitím dostupných vstupů a výstupů
  - Vytvořením vlastních vstupů a laděním programu
- Penalizace za překročení počtu uploadů
- Detekce plagiátů
- Porozumění kódu a kontrola možných stavů
  - Pro každou funkci nebo načtení vstupu od uživatele analyzujte možné vstupní hodnoty nebo návratové hodnoty funkcí
  - Pokud je z hlediska funkčnosti vstup nebo návratová hodnota zásadní, proveďte kontrolu vstupu a/nebo příslušnou akci, např. vypsání hlášení a ukončení programu

Např. očekávaný vstup je číslo a uživatel zadá něco jiného.

Hodnocení

Zdroj bodů	Maximum	Nutné minimum
Domácí úkoly	50	35
Test v semestru	10	
Zápočtový test	35	15
Implementační test	15	-
Součet	110	

- Za práci v semestru je třeba získat nejméně **35 bodů**, všechny domácí úkoly musí být odevzdány a to nejpozději do 12.1.2020 ve 23:59 CET!
- Implementační test – schopnost pochopit problém a napsat krátký program (cca 4 hodiny)

Klasifikace

Klasifikace	Bodové rozmezí	Slovní hodnocení
A	≥ 90	výborně
B	$80 - 89$	velmi dobře
C	$70 - 79$	dobře
D	$60 - 69$	uspokojivě
E	$50 - 59$	dostatečně
F	< 50	nedostatečně

Přehled přednášek

1.	Informace o předmětu, úvod do programování	22.9.	
2.	Základy programování v C, překlad, chyby	29.10.	HW01
3.	Reprezentace dat v paměti, základní řídící struktury	6.10.	HW02
4.	Řídící struktury, výrazy, funkce	13.10.	HW03
5.	Strukturované datové typy, ukazatele	20.10.	HW04
6.	Textové řetězce	27.10.	HW05
7.	Práce s pamětí, zásobník, halda	3.11.	HW06
8.	Vnitřní reprezentace datových typů	10.11.	HW07
9.	Spojové struktury, abstraktní datový typ	24.11.	HW08
10.	Generický datový typ, ukazatele na funkce	1.12.	HW09
11.	Studentská volba 2	8.12.	HW10
12.	Studentská volba 1	15.12.	HW11
13.	Zápočtový test	5.1.	

Studentská volba

1. Programování v Pythonu
   - Seznámení s jazykem a klíčovými knihovnami (numpy, ...)
   - Praktické příklady: výpočty, kreslení grafů, zpracování textových souborů
2. Pokročilé partie programování v C
   - Paralení programování, paralelní výpočty a synchronizační primitiva (semafory, zprávy a sdílená paměť)
   - Vícevláknové programování, modely aplikací, POSIX vlákna C11 vlákna
3. Úvod do programování v C++
   - Stručné seznámení s jazykem
   - Velmi lehký úvod do objektového programování
4. Velmi jemný úvod do embedded programování
   - Seznámení s kitem Nucleo
   - Jednoduché programy v prostředí MBED, LEDky, sériová komunikace, ...

O náplni 11. a 12. přednášky rozhodne hlasování, které bude spuštěno ve třetím týdnu.

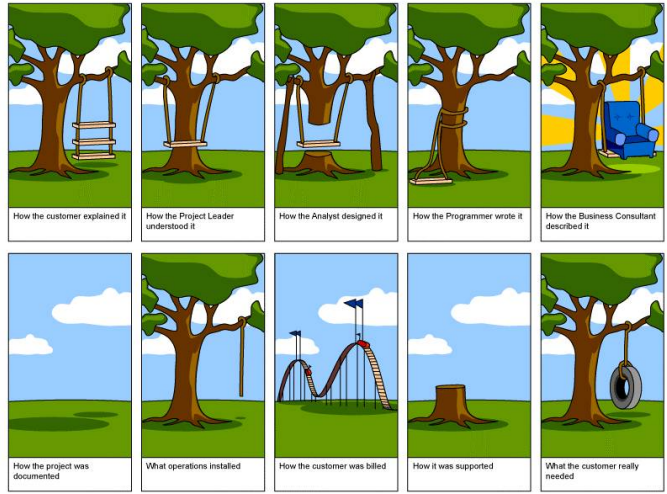
Část II

O programování

II. O programování

Řešení problémů

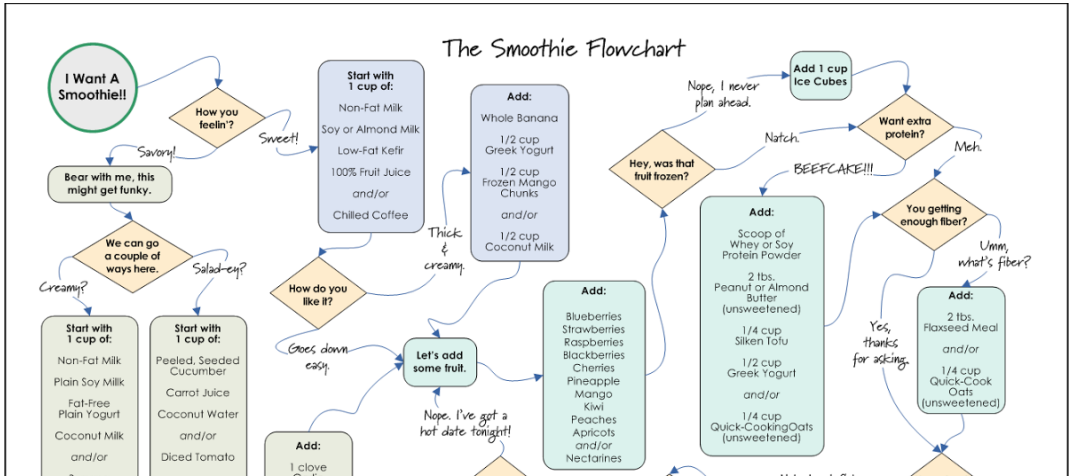
1. formulace problému
2. analýza možných řešení
3. návrh algoritmu
4. implementace
5. ověření funkčnosti
6. optimalizace
7. oprava chyb
8. údržba
9. dokumentace



Co je to program?

- Program je **recept** – posloupnost kroků (výpočtů), popisující průběh řešení nějakého problému pomocí dostupných prostředků – programovací prostředí, počítač, ...

Receptu budeme říkat algoritmus.



Co je to 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.

https://www.youtube.com/watch?v=Ct-l0OUqmyY

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

Zápis algoritmu

- Existují 4 hlavní způsoby, jakými lze algoritmus popsat:
  - slovně – vyjádříme slovně postup řešení a jednotlivé kroky
  - graficky – použití vývojových diagramů a struktogramů
  - matematicky – jednoznačnný popis matematickou konstrukcí (např. rovnicí)
  - programem – kroky algoritmu jsou popsány programovacím jazykem
- Návrhy algoritmů:
  - shora dolů – problém rozdělíme na několik podúloh, které řešíme
  - zdola nahoru – z triviálních úloh skládáme vyšší úlohy
  - kombinace obou metod

V praxi vždy záleží především na komplexnosti a povaze řešeného algoritmus, který postup bude nejlepší aplikovat.

Základní složky programů a algoritmů

- Programy zpravidla transformuje množinu vstupních dat na množinu dat výstupních
- Základní složky programů

  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 programu

  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).

Jak začít?

Jednoduché algoritmy, grafické programování

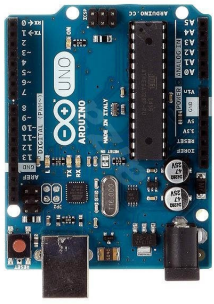
- Scratch – MIT Media Lab
- Angry Birds
- Code with Anna and Elsa
- Minecraft

Programovací jazyk Karel

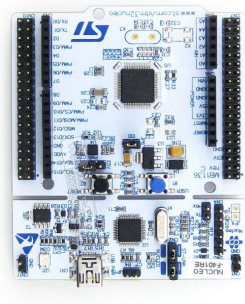
- Pohyb robota po čtvercové sítí
- Richard E. Pattis, Karel The Robot: A Gentle Introduction to the Art of Programming, Stanford, 1981
- Online: Stanford , Oldřich Jedlička

Další zajímavé programovací jazyky pro výuku programování např. zde .

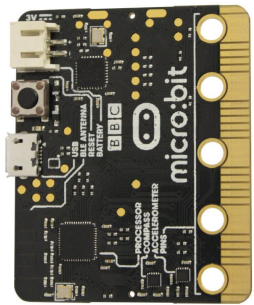
Programování může být skvělá zábava



Arduino
Open Source
Procesory AVR



Nucleo
ST Microelectronics
Procesory ARM



BBC Micro:bit
Open Source
Procesory ARM

II. O programování

Programovací jazyk? C

... 1960s

Bell Labs

Ken Thompson

Space Travel

GE 635

PDP-7

Unix

Dennis Ritchie

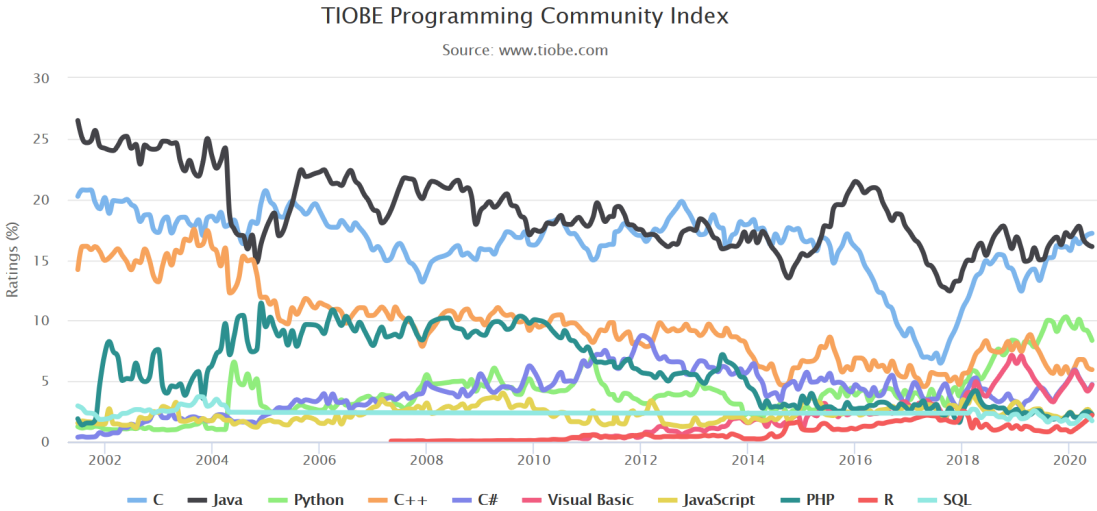
B → C

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
C
Java
Python
C++
C#
Visual Basic
JavaScript
PHP
R
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

II. O programování

První program – 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!
```

První program – 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!
```

První program – 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!
```

První program – 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!
```

První program – 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!
```

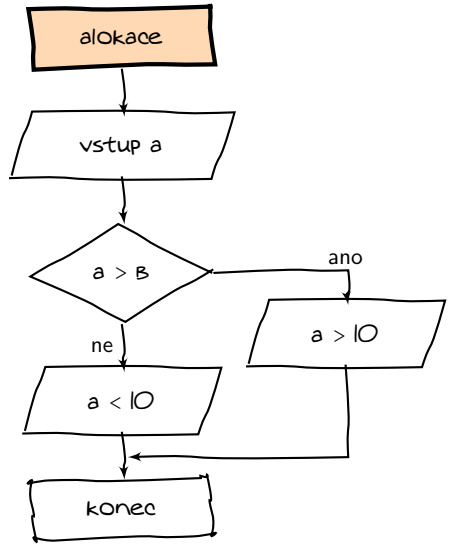
II. O programování

Druhý program – 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
```

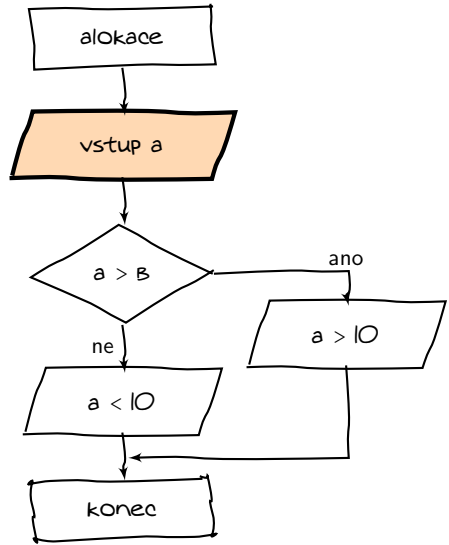


Druhý program – větvení

```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
```

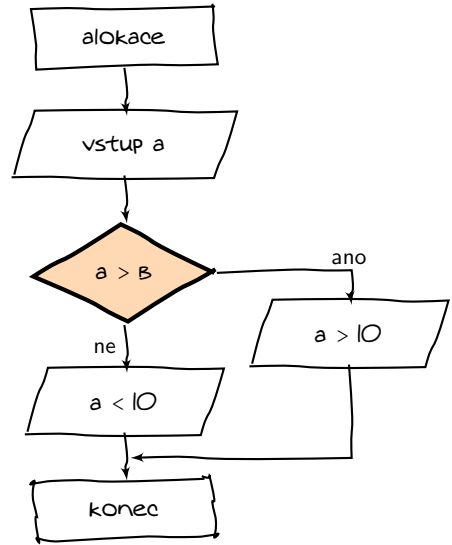


Druhý program – větvení

```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
```

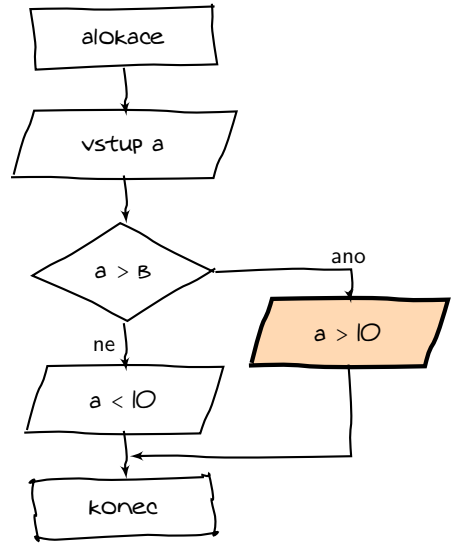


Druhý program – větvení

```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
```

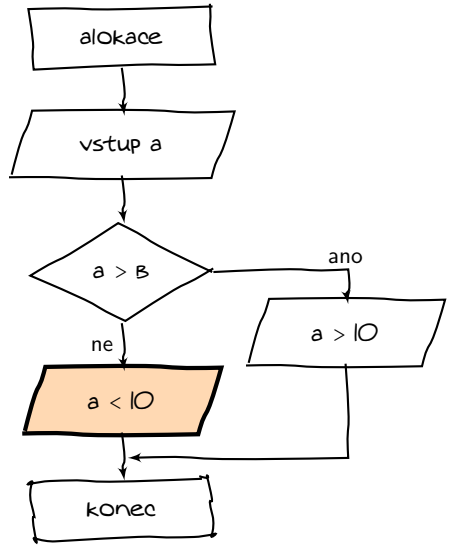


Druhý program – větvení

```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
```

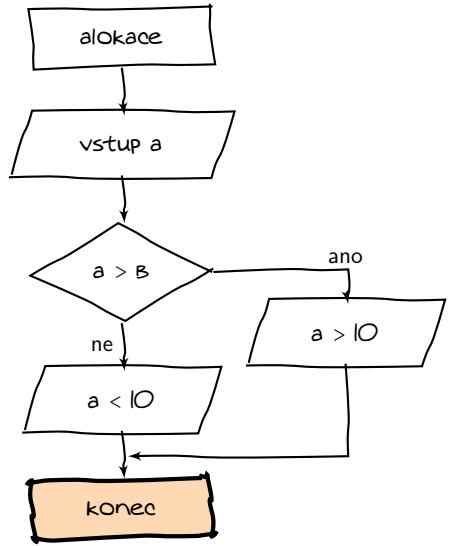


Druhý program – větvení

```
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
```



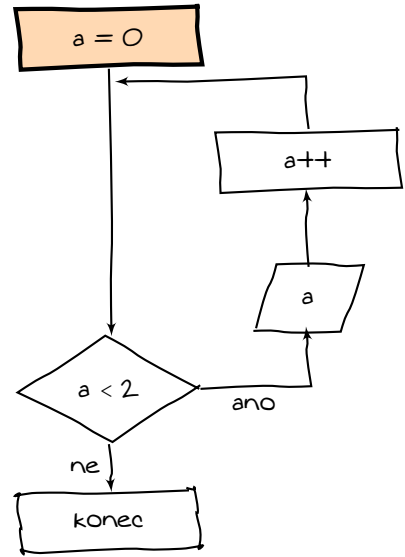
II. O programování

Třetí program – 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
```

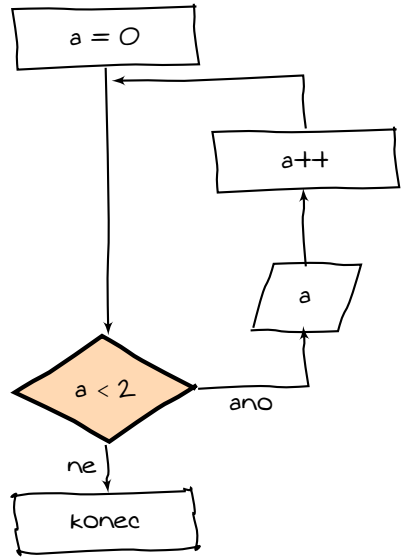


Třetí program – cyklus

```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
```

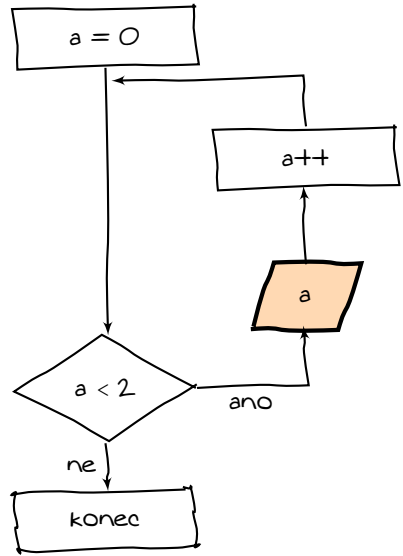


Třetí program – cyklus

```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
```

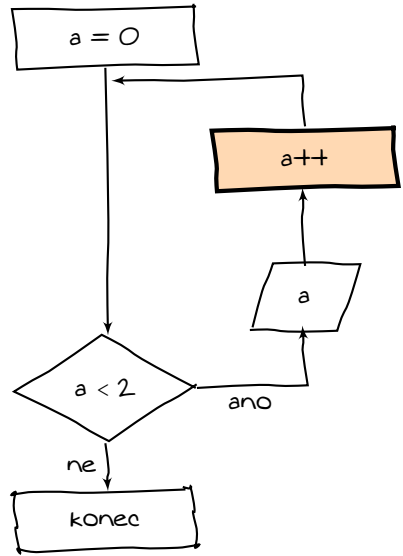


Třetí program – cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;

  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }

  return 0;
}
```

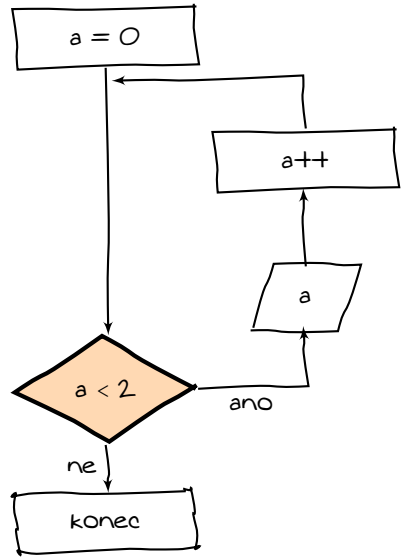
- Na displej se vypíše řada čísel

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



Třetí program – 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

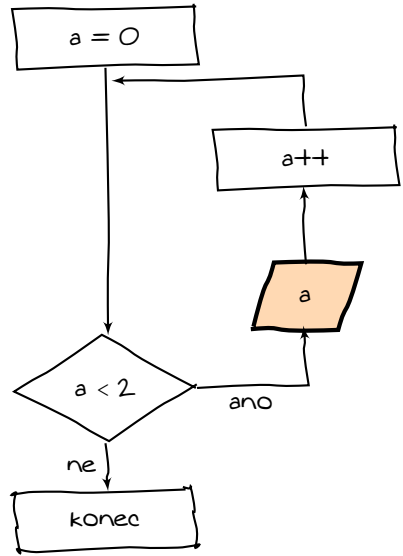
```
$ ./a.out
0 1
```

Třetí program – 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
```

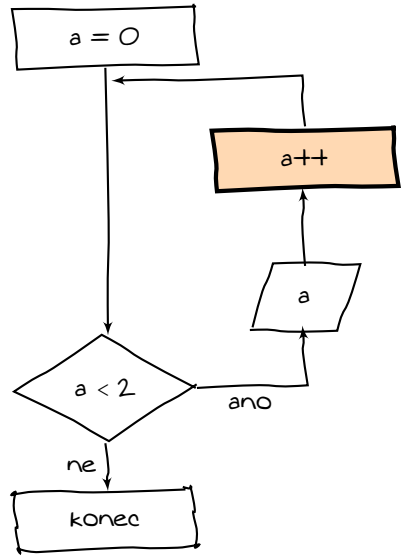


Třetí program – cyklus

```c
#include <stdio.h>

int main()
{
  int a = 0;

  while (a < 2)
  {
    printf ("%d", a);
    a++;
  }

  return 0;
}
```

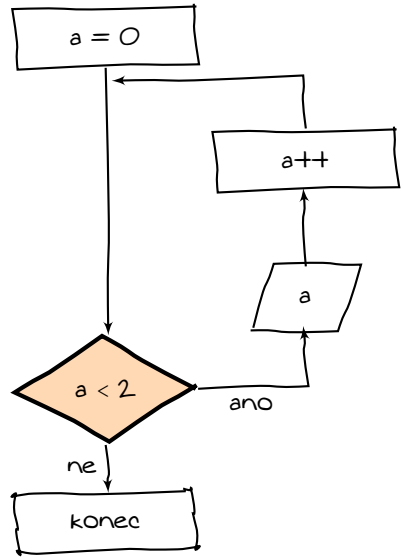
- Na displej se vypíše řada čísel

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



Třetí program – 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

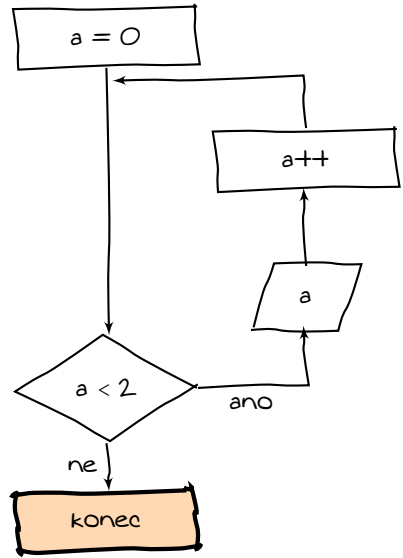
```
$ ./a.out
0 1
```

Třetí program – 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
```



II. O programování

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`