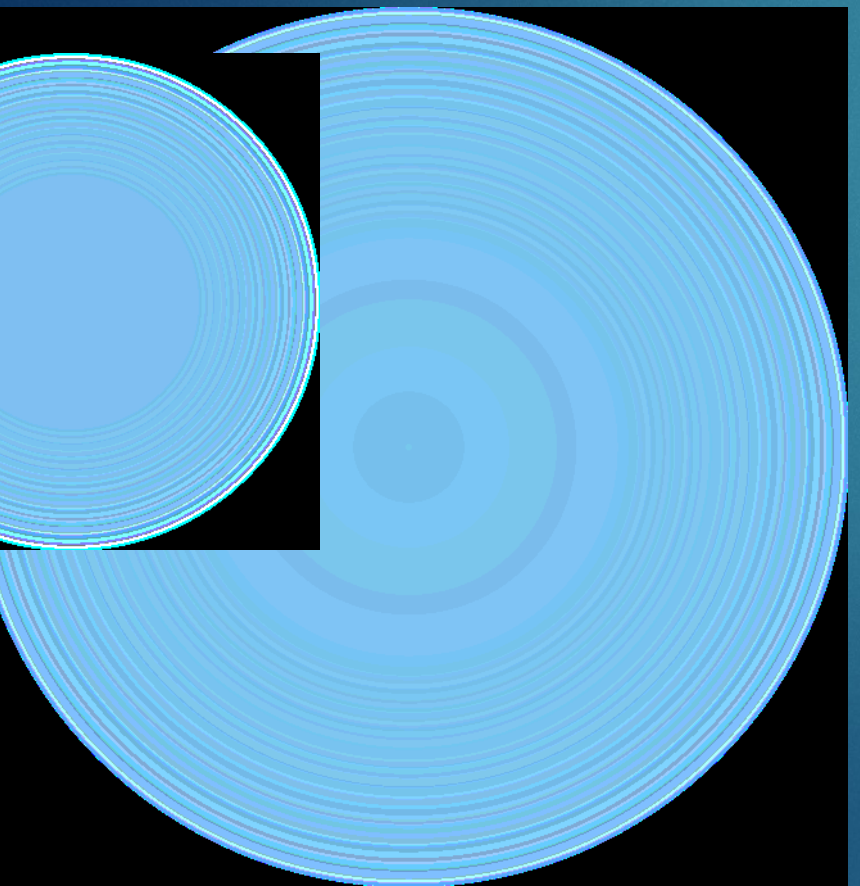


Od Scratche k něčemu složitějšímu

- Nižší programovací jazyk
  - Malá míra abstrakce oproti tomu, jak funguje procesor počítače
  - Těsně spjaté s hardwarem
  - Nutná technická znalost fungování hardware
  - Závislý na platformě
- Vyšší programovací jazyk
  - Velká míra abstrakce
  - Přiblížení zápisu tomu, jak uvažuje člověk
  - Scratch, Python, BASIC, Java, PHP, …
  - C někde uprostřed – umožňuje nízko úrovňový přístup k paměti aj.

Objektové vs sekvenční

- Objektové programování
  - Komunikace jednotlivých prvků – objektů
  - Objekty jsou uzavřené, navenek poskytují jen metody
  - Nemusíme vědět, jak objekty fungují
  - Třídy – skupiny objektů sdílející stejné vlastnosti
  - Vhodné pro aplikace s GUI, hry, …
  - Objekt C, většina jazyků kombinuje obojí přístup
- Sekvenční programování
  - Kód je vykonáván řádek po řádku
  - Funkcionální programování – kód obsahuje funkce (přeskakuje)
  - Větvení a cykly
  - Vhodný pro vědecké aplikace, simulace
  - C

Kompilované/interpretované

- Interpretovaný jazyk
  - Není potřeba překlad do strojového kódu
  - Potřebují speciální software – interpreter
  - Méně přenositelné, pomalejší, ale jednodušší pro uživatele, snadný debugging
  - Přímo interpretovaný/přeložen do mezikódu/částečně interpretovaný
  - Někdy se nazývají skriptovací jazyky
  - Python, Shell, Cmd, Wolfram, …
- Kompilovaný jazyk
  - Zdrojový kód je nejprve přeložen do strojového kódu a poté proveden celý najednou
  - Podstatně výkonnější, přenositelné, nutná dobrá znalost jazyka
  - C, C++

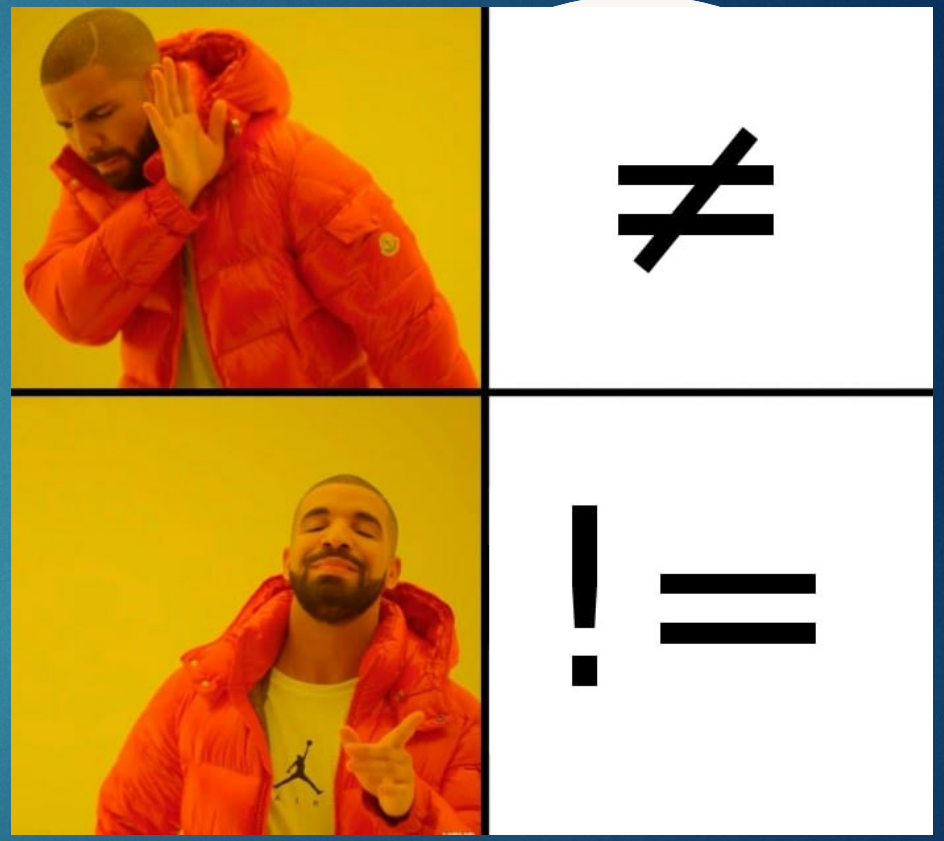
Hello Scratch

Hello world v „Assembleru“

```
[org 100h]
[bits 16]
jmp START

; Nastavit pozici kurzoru
; IN: dl = x, dh = y
curto:
xor bh,bh
mov ah,2
int 10h
ret

; Napsat barevne znaky, ale neposouvat kurzor
; IN: al = char, bl = color, cx = count
putchar:
xor bh,bh
mov ah,9
int 10h ret

; Napsat znak a posunout kurzor
; IN: al = char
wrchar:
xor bh,bh
mov ah,0Eh
int 10h
ret

; Cist klavesu s cekanim ;
OUT: al = ASCII code || 0, ah = scan code
inkey:
mov ah,0
int 16h
Ret
; Napsat textovy retezec ukonceny binarni nulou
; IN: ds:si -> null_terminated_string
writez:
```

Hello world v C

```
#include <stdio.h>

int main (){
        printf („Hello, world!\n“);
        return 0;
}
```

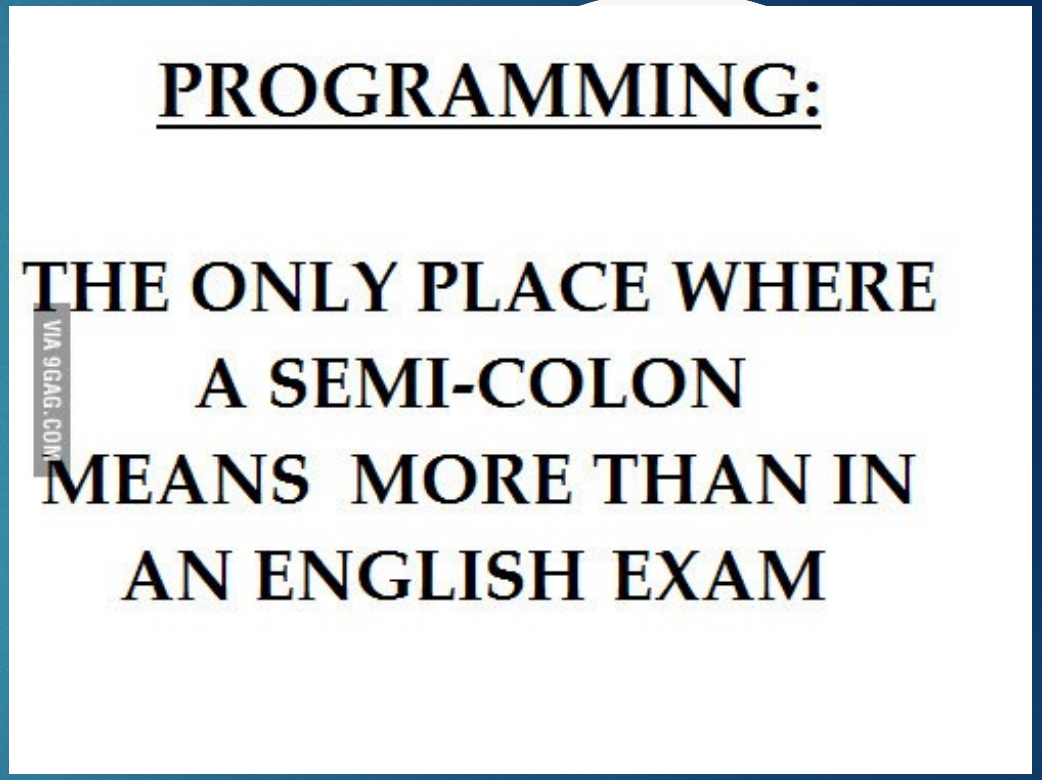


C

PROGRAMOVACÍ JAZYK

Klíčová slova

- Slova a symboly mající určitý význam v daném programovacím jazyce
- Většinou je jejich použití jinde zakázáno
- Označení datového typu, funkce knihoven, operátory, terminální symboly, direktivy preprocesoru, …
- Př. int, include, double, const, main, printf, sin, +, ==, ;, #, …



Operátory

Unitární

+,- unární plus a mínus,
& reference (získání adresy objektu)
* dereference (získání hodnoty objektu dle adresy)
! logická negace
~ bitová negace
++, -- inkrementace a dekrementace hodnoty
(typ) přetypování
sizeof() operátor pro získání délky objektu nebo typu!

Binární

= přiřazení
+,-,*,/ plus, mínus, krát, děleno
% zbytek po celočíselném dělení (modulo)
&& logické AND
|| logické OR
. tečka, přímý přístup ke členu struktury
-> nepřímý přístup ke členu struktury

Relační

< menší než
> větší než
<= menší nebo rovno
>= větší nebo rovno
== rovnost
!= nerovnost

Funkce

- Urychlení běhu programu, usnadnění orientace ve zdrojovém kódu, zkrácení kódu
- Funkce – navrací hodnotu
- Procedura – nevrací nic
- Skladba:

datový typ který funkce vrací jméno funkce (**datový typ vstupní proměnné** vstupní proměnná){

to, co funkce provádí;

}

Cykly, podmínky

- Cykly
  - Usnadnění kódu – počítač dokáže velmi rychle provádět milióny úkolů
  - for (i = 0; i < 10; i++){
Tělo cyklu; }
  - While (true){
    tělo cyklu; }
- Podmínky
  - Větvení programu
  - Rozhodování pomocí logických operátorů, porovnávání hodnot proměnných
  - if (true) …
  - if (true) …. else …
  - if (true) … else if (true) … else
  - Switch - case

```
if (5 == count)
```

Proměnné

- V některých jazycích není nutné určovat typ (Python) – možnost chyb
- Datový typ – jaké místo má být přiřazeno
- Int, float, double, char, …
- Konstanta – urychlení běhu programu, jiný způsob ukládání do paměti
  - Přímé zadání hodnoty
  - Konstantní proměnná
  - Preprocesorová proměnná

Pointery

- Proměnná je místo v paměti počítače, je jí přidělena adresa
- Pointer (ukazatel) ukazuje na adresu proměnné
- K hodnotě na dané adrese přistupujeme pomocí *
- V jazyce C hojně používané, ovšem často vede k chybám
- Pointerová aritmetika

```
int a;

a = 56;

printf("Proměnná a s hodnotou %d je v paměti
uložená na adrese %p", a, &a);
```

Pointery

```
int i;          // definice proměnné i datového typu int
int a[3];       // definice proměnné a, pole typu int se třemi prvky
int *p;         // definice proměnné p, ukazatele na datový typ int
p = &i;         // hodnota pointeru je nastavena na adresu proměnné i
*p = 3;         // do paměti na adresu odkazovanou ukazatelem p se uloží
                                        hodnota 3
p = &(a[2]); // hodnota pointeru je nastavena na prvek pole a s indexem 2
p = p - 2;      // hodnota pointeru je nastavena na prvek pole a s indexem 0
                                        (tj. první prvek pole)
*p = 5;         // do paměti na adresu odkazovanou p se uloží hodnota 5 (tedy
                                        první prvek pole má tuto hodnotu)
```

Pole

- Vícerozměrná struktura proměnných (libovolná dimeze)
- Přístup pomocí ukazatelů
- Daná velikost při deklaraci
  - Datovy_typ nazev [velikost];
- Alokace paměti – pro větší pole, zadávám velikost v bytech (malloc, calloc, free – stdlib.h)
- Struktury (c++) – více typů proměnných v 1 „objektu“
- Vector – proměnná velikost

```
float vector[3];
double matrix[2][5];
int **pole;
pole = (int**) malloc(m*sizeof(int));
for (i = 0; i < n; i++){
        pole[i] = malloc(3*sizeof(int));
        }
free (pole);
```

Zdrojový kód v C

- Tělo funkce nebo podmínky uzavřeno do složených závorek {}
- Každý příkaz ukončen středníkem; obvykle i konec řádku
- Parametry funkce v kulatých závorkách
- Pozice/velikost pole v hranatých závorkách
- Komentáře odděleny // na 1 řádku, případně uzavřeny do /* jako blok */

```
int array[10];  \\ definice pole integerů o velikosti 10
for (i = 0; i < 10; i ++){
        array[i] = array [i -1] + array[0];
        /*
        my_func (array[i]); */
}
```

Čitelnost zdrojového kódu

- Správné formátování a volba názvů proměnných a funkcí/procedur usnadňuje orientaci v kódu
- IDE pomáhají barevným odlišováním klíčových slov
- Case sensitive, klíčová slova psána malými písmeny
- Tělo funkce/cyklu odsazeno tabulátorem
- Mezery mezi operátory
- Volné řádky oddělují jednotlivé bloky programu
- Identifikátor – název proměnné, funkce
  - Nesmí mít číslo a _ na začátku
  - Bez diakritiky

„Always code as if the guy who end up maintaining your code will be a violent psychopath who knows where you live.“

Sestavení programu

- KOMPILÁTOR
  - Překládá zdrojový kód napsaný programátorem to objektového kódu, kterému rozumí počítač
  - Alokuje paměť pro proměnné
- PREPROCESOR
  - V C/C++
  - Direktivy – instrukce pro kompilátor
  - Preprocesorové proměnné – např. konstanty
  - Začínají #
- LINKER
  - Doplňuje do objektového kódu odkazy na knihovny a vytváří tak finální strojový kód

Hello, world!

```
#include <stdio.h>

int main (){
        printf („Hello, world!\n“);
        return 0;
}
```

```
01111111 01000101 01001100
01000110 00000010 00000001
00000001 00000000 00000000
00000000 00000000 00000000
00000000 00000000 00000000
00000000 00000010 00000000
00111110 00000000 00000001
00000000 00000000 00000000
10110000 00000101 01000000
00000000 00000000 00000000
00000000 00000000 01000000
00000000 00000000 00000000
00000000 00000000 00000000
00000000 11010000 00010011
00000000 00000000 00000000
00000000 00000000 00000000
00000000 00000000 00000000
```