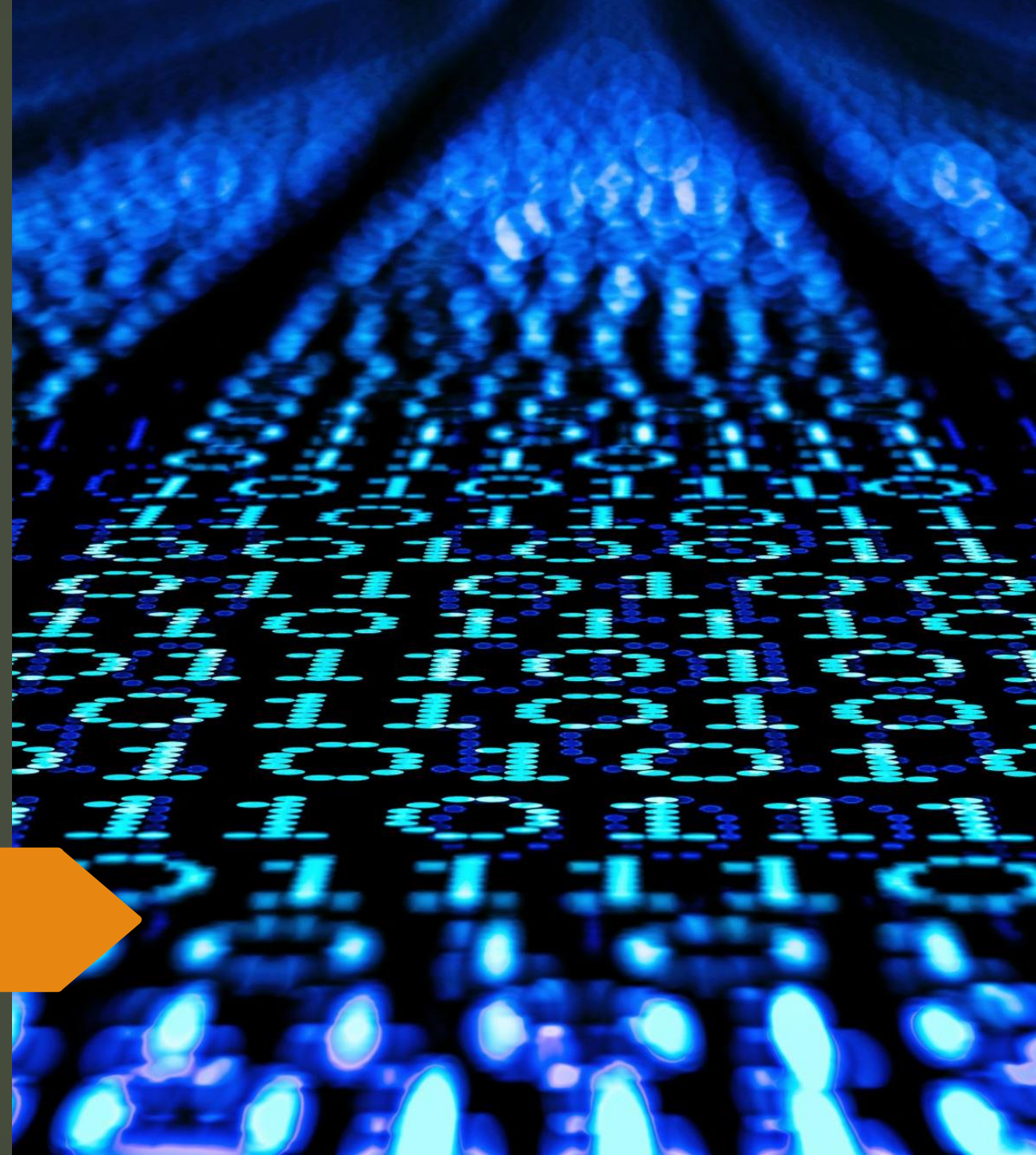


# Programování

Proměnná, konstanta, datové typy






# Pojmy k vysvětlení

## Proměnná

- ▶ Lokální
- ▶ Globální
- ▶ Konstanta

## Datový typ

- ▶ Logický
- ▶ Číselný
  - ▶ Celočíselný
  - ▶ Reálný
- ▶ Znakový



# Proměnná, konstanta

- ▶ = pojmenované místo v paměti počítače, která drží určitou informaci
- ▶ Typ informace v proměnné je určen datovým typem
- ▶ Hodnota v proměnné se může za běhu programu měnit
- ▶ Pokud je hodnota proměnné zafixována a nelze ji upravit (nemění se) říkáme takové proměnné **konstanta**
  
- ▶ Pojmenováním proměnné vytváříme referenci na danou hodnotu
- ▶ Pro pojmenování proměnných je nutné dodržovat jistá pravidla odvíjející se od programovacího jazyku

# Práce s proměnnou

- ▶ Deklarace proměnné
- ▶ Proces při kterém se v paměti počítače vyhrazuje prostor pro novou proměnnou
- ▶ Skládá se z určení datového typu a pojmenování  
**double nova\_promenna;**
- ▶ Deklarovaná proměnná je často inicializována na výchozí hodnotu
- ▶ Inicializace proměnné
- ▶ Přiřazení počáteční hodnoty proměnné (využití přiřazovacího operátoru =)  
**double nova\_hodnota = 27,31;**
- ▶ Vložení hodnoty do proměnné
- ▶ Vložení hodnoty do proměnné v dalších krocích programu provádíme s využitím přiřazovacího operátoru =





# Lokální a globální proměnná

- Určování viditelnosti proměnných v těle programu
- **Lokální proměnná** je viditelná uvnitř těla funkce případně samotné třídy
- Se vznikem funkce vzniká i proměnná a na konci funkce se informace o proměnné ztrácí
- Obdobné je to i u proměnných třídy struktury – lze dodatečně upravit modifikátory viditelnosti (private, packaged)
- **Globální proměnná** je přístupná z libovolné části programu
- Modifikátor viditelnosti public

# Konstantní proměnná

- Proměnná, která je po své inicializaci neměnná
- Klíčové slovo **const**
- Pro přehlednost v kódu se doporučuje konstantní hodnoty pojmenovávat pouze velkými písmeny

**int const KONSTANTNI\_CISLO = 123;**

- Při pokusu o změnu hodnoty v konstantní proměnné jsme kompilátorem upozornění na chybu
- Některé předdefinované knihovny, třídy mají své vlastní proměnné, které lze využít např. **Math.PI**



# Datový typ proměnné

- ▶ Určuje jaký typ informace je v proměnné uložen
- ▶ V případě, že se pokusíme do takové proměnné vložit hodnotu jiného typu, jsme upozorněni kompilátorem
- ▶ Implicitní konverze – neztrácíme informaci o hodnotě
- ▶ Explicitní konverze – ztráta informace o hodnotě
- ▶ Ke konverzi můžeme využít přetypování, parsovací funkce nebo funkce pro konverzi (Convert.ToX)



# Logické datové typy

- Jediným logický datovým typem je **bool**
- Může nabývat pouze hodnot **true** nebo **false**
- Logický datový typ je požadován jako parametr podmínek v cyklech **while**, **do-while**, případně u rozhodovacího **if-else** bloku
- Logický datový typ je výsledek výrazů, u kterých lze rozhodnout o jejich pravdivosti – logické operátory
- Logický součin, součet, negace, ekvivalence, porovnávání





# Číselné datové typy

- ▶ Datové typy, které drží informaci o čísle
- ▶ **Celočíselné datové typy**
  - ▶ byte
  - ▶ short
  - ▶ int
  - ▶ long
- ▶ **Reálné datová typy**
  - ▶ float
  - ▶ double
- ▶ Jednotlivé datové typy jsou od sebe rozlišné svou bitovou velikostí a tedy i rozsahem / přesností

# Znakové datové typy

- Jediným znakovým typem je **char**
- V tomto datovém typu lze uchovat jeden znak
- V kódu odlišujeme hodnoty typu char pomocí jednoduchých uvozovek
- Jednotlivé znaky jsou reprezentovány pomocí Unicode
- Char je často využíván pro následnou práci s řetězcí – iterování v řetězcí

**char znak = 'X';**

# Přehledová tabulka rozsahů datových typů a jejich bitová velikost

Datový typ	Bitová velikost	Rozsah hodnot
byte	8 bitů	0 až 255
short	16 bitů	-32 768 až 32 767
int	32 bitů	-2 147 483 648 až 2 147 483 647
long	64 bitů	-9 223 372 036 854 775 808 až 9 223 372 036 854 775 807
float	32 bitů	$1,5 \times 10^{-45}$ až $3,4 \times 10^{38}$
double	64 bitů	$5,0 \times 10^{-324}$ až $1,7 \times 10^{308}$
bool	1 bit	true nebo false
char	16 bitů	Unicode znaky