Jazyk C

- Je to vyšší kompilovaný programovací jazyk. Kompilovaný znamená, že zdrojový kód musí být nejprve kompilován do strojového kódu, aby bylo možné soubor s instrukcemi spustit.
- Zdrojový kód, kde se nachází příkazy se dělí na dva konkrétní druhy:
  - Syntaxe = tvar příkazu + parametrů („jak se to napíše“)
  - Sémantika = význam příkazu + parametrů („co to dělá“)
- Case sensitive – rozlišuje malá a velká písmena v názvech proměnných, konstant, funkcí a hodnot. (pinMode (13, OUTPUT);)

Datové typy + příklady k nim

Datové typy určují:

- množinu hodnot (minimum a maximum)
- povolené operace
- velikost místa v paměti (kolik Bajtů)

Celá čísla: deklarují se slovem **int** a pro výpis se značí **%d**

```
int pocet = 13;

printf("Počet: %d", pocet);
```

Desetinná čísla: deklarují se slovem **float** a pro výpis se značí **%f**

```
float pocet = 13.3;

printf("Počet: %f", pocet);
```

Znaky: deklarují se slovem **char** a pro výpis se značí **%c**

* U znaků se hodnota vždy dává do apostrofů!

```
char ano = 'A';

printf("Znak: %c", ano);
```

Textový řetězec: deklarují se slovem **char []** a pro výpis se značí **%s**

```
char slovo [21];

printf("Slovo: %s\n", slovo);
```

Proměnná, konstanta: pojmenované místo v paměti (datový typ, název-identifikátor

Za běhu programu může měnit hodnotu	Za běhu programu nemůže měnit hodnotu

Proměnná – obsahuje datový typ, název a hodnotu - může být doplněna za běhu

① Deklarace proměnné

Datový typ + název --> **int age;**

Inicializovaná proměnná --> **int vek = 25;**
(Doplnění konkrétní hodnoty)

② Použití proměnné

```
int age;
age = 7;
printf("%d", age);
```

Konstanta – obsahuje datový typ, název a hodnotu - musí být definována na začátku

① Deklarace konstanty

const datový typ + název = hodnota --> const int year = 2023;

② Použití konstanty

```
const int year = 2023;
```

Při deklarace proměnné, mohu zapsat na začátku pouze int age; a v polovině programu doplnit její hodnotu.

Při deklarace konstanty, musím na začátku uvést i hodnotu, protože ji v polovině měnit nemohu!!

Struktura programu v jazyku C

Hlavní program

Hlavní program v jazyce C je část kódu, která začíná funkcí main(void) / main(). Tato funkce je vstupním bodem každého programu v C — když se spustí program, jeho běh vždy začíná právě ve funkci main(void) / main().

Začátek se zapisuje: int main(void) nebo pouze jako int main()

Konec se zapisuje: return 0;

```
int main(void) {

                                    Hlavní program

  return 0;
}
```

Funkce

① **Deklarace funkce** – vždy nad hlavním programem!

a) Funkce bez parametrů

```
char getOneChar () {     // V závorkách není parametr
  return 'C';            // Návratová hodnota
}
```

```
int getOneNumber () {    // v závorkách není parametr
  return 13;             // Návratová hodnota
}
```

b) Funkce s parametry – hodnotám v závorkách (A, B) se říká formální parametry!

```
int addNumbers (int A, int B) {     // Hodnoty v závorkách
  return A + B;                     // Návratová hodnota
}
```

② **Volání funkce –** při volání funkce jsou uvnitř závorek skutečné parametry! (deklarované nad programem)

```
int Vysledek;                       // Deklarace proměnné
int Num1 = 7;                       // Deklarace proměnné
int Num2 = -5;                      // Deklarace proměnné

int addNumbers (int A, int B) {     // Funkce s form. par.
  return A + B;                     // Návratová hodnota
}

int main(void) {                    // Začátek hl. prog.
  printf("Napiš dvě čísla: ");      // Zápis čísel
  scanf("%d %d", &Num1, &Num2);     // Čtení z klávesnice

  Vysledek = addNumbers (Num1, Num2);     // Volání funkce
                                          // Skutečné par.
```

```
  return 0;                              // Ukončení
}
```

Globální a lokální proměnné

Globální proměnná

- jsou deklarovány mimo blok složených závorek {}
- platí v hlavním programu i ve všech funkcích (pro celý program)

Lokální proměnná

- jsou deklarovány uvnitř funkce, v bloku složených závorek {}
- platí pouze uvnitř funkce, kde jsou deklarovány

Formální parametr funkcí se uvnitř funkce chová jako lokální proměnná

! Uvnitř funkce neměnit hodnotu globální proměnných !

```
#include <stdio.h>                       // Knihovna

int n1, n2, RES, RES1;                   // Globální proměnné

int f_soucetCisel(int a, int b) {   // Lokální proměnné
  RES1 = (a + b);
  return RES1;
}

int main(void) {                         // Začátek programu
  printf(„Napiš dvě čísla: „);           // Zobrazení zprávy
  scanf(„%d %d", n1, n2);                // Čtení z klávesnice

  RES = f_soucetCisel(n1, n2);           // volání funkce
  printf(„Součet čísel je: %d, RES); // Výpis výsledku

  return 0;                              // Ukončení programu
}
```

Logické operátory

OR --> || if ((Num1 < 0) || (Num2 < 0)) alespoň jeden z nich

AND --> && if ((Num1 < 0) && (Num2 < 0)) obě současně

Srovnání jazyku C s JavaScriptem

- Jazyk C se musí kompilovat (zdrojový kód do strojového). JavaScript je možno interpretovat i kompilovat v průběhu pomocí (JIT – Just-In-Time) kompilátoru.
- Jazyk C vyžaduje po programátorovi přiřazování a odebírání prostoru v paměti. JavaScript dělá tuto akci automaticky.
- Kód jazyka C musí být překompilovaný při přesunu do jiného procesoru. JavaScript nemusí být.
- C nabízí přímou kontrolu podprocesů, mezitím JavaScript vybízí uživatele k práci s více úlohami tím, že rozdělí úkoly do asynchroních funkcí, které jsou zavolány, když jsou data připravena.

Praktický příklad – Replit C-25

```c
//·Nadeklaruj·funkci·f_najdiMaximum()·datového·typu·int
//·Funkce·bude·mít·tři·formální·parametry·"a,·b,·c"·datového·typu·int
//·Funkce·vrátí·návratovou·hodnotu·maximum·z·těchto·tří·čísel
//·Napiš·hlavní·program,·ve·kterém·uživatel·zadá·postupně·tři·celá·čísla,·zavolej·tuto·funkci·a·předej·jí·tyto·skutečné·parametry
//·Funkce·vrátí·do·hlavního·programu·hodnotu·maxima,·kterou·v·hlavním·programu·vypíšeš

#include·<stdio.h>

int·cislo1,·cislo2,·cislo3,·maximum,·soucet;

int·f_najdiMaximum(int·a,·int·b,·int·c)
{
··if(a·>·maximum)·maximum·=·a;
··if(b·>·maximum)·maximum·=·b;
··if(c·>·maximum)·maximum·=·c;·
··return·maximum;
}

int·main(void)
{
··printf("Napiš·tři·čísla:·\n");
··scanf("%d·%d·%d",·&cislo1,·&cislo2,·&cislo3);

··printf("\n");

··soucet·=·f_najdiMaximum(cislo1,·cislo2,·cislo3);
··printf("Největší·číslo·je:·%d",·soucet);

··return·0;
}
```