Funkce

Slouží k logickému členění kódu.

Každá funkce má svou *definici*, která obsahuje

- typ výsledku (funkční hodnoty),
- jméno funkce,
- seznam *formálních parametrů* s jejich typy a jmény,
- *tělo* (ukončené příkazem `return`).

Typ, jméno a seznam formálních parametrů tvoří *hlavičku* funkce.
Tělo funkce popisuje algoritmus výpočtu.
Hlavička uvedená bez těla se nazývá *prototyp* funkce.

Volání funkce neboli její *aplikace* na argumenty se v kódu může vyskytovat (a zpravidla vyskytuje) víckrát. Volání se skládá

- ze jména funkce
- a seznamu *skutečných parametrů* neboli *argumentů* v závorkách; seznam může být i prázdný.

Čisté funkce

Funkce v matematickém smyslu.

- Výsledek závisí jen na argumentech,
- Jednoznačné: volání se stejnými argumenty vede ke stejnému výsledku.
- Totální: volání na každých argumentech skončí úspěšně.
- Neprovádějí I/O.
- Nemění stav výpočtu.
- Pracují-li se stavem, tak jen s lokálními (zásobníkovými) proměnnými.
- Pokud si alokují paměť na hromadě, vždy ji nakonec samy uvolní.

Příklady:

`int abs(int n)`	absolutní hodnota
`char tolower(char c)`	převod písmene na malé
`int strlen(char* s)`	délka řetězce
`double log(double x)`	přirozený logaritmus; čistá, až na úplnost

Funkce v C

Mohou *záviset* na stavu výpočtu a na prostředí, v němž program běží: globální data (statická i dynamická), vstupy, služby OS.

Mohou *měnit* stav: výstupy, globální data, alokace/uvolňování paměti. Funkce, které mění stav výpočtu, se také nazývají *procedury*.

Často používané jsou funkce, které mění hodnotu svého parametru.

Příklady:

`double rand(void)`	závisí na svém stavu
`void sort(int n, int a[])`	mění obsah 2. parametru
`int getchar(void)`	závisí na vstupu a spotřebovává ho
`void printspaces(int n)`	provádí výstup
`void* malloc(int size)`	alokuje paměť na hromadě

Doporučení pro psaní funkcí

- Pokud je to možné, preferujeme čisté funkce.
- Ostatní funkce / procedury by měly mít *minimální* rozhraní.
- Jméno funkce má vystihovat její sémantiku.
- Každou definici komentujeme a v komentáři přesně popíšeme interakci funkce s okolím:
  - na čem funkce závisí,
  - co vrací,
  - definiční obor, tj. přípustné hodnoty parametrů,
  - které své parametry mění,
  - jak ovlivňuje stav: hromadu, I/O, OS, globální proměnné.

Největší číslo v poli

Implementujte hledání největšího čísla v poli jako funkci. Délka pole je jejím prvním parametrem, pole druhým.

https://github.com/li-ska/izp-cv

```
#include <stdio.h>

int maximum (int n, int a[])  {
      /*  DOPLŇTE  */
}

int main (void) {
  int n;
  int a[100];
  do { printf("Length (positive): "); scanf("%d", &n);
  } while (n<1);
  printf("List of %d ints: ", n);
  for (int i=0; i<n; i++) scanf("%d", &a[i]);
  printf("Maximum: %d\n", maximum(n,a));
  return 0;
}
```

Vyhledávání v řetězci

Napište funkci `int match(char pattern[], char text[])`, která najde pozici prvního výskytu vzorku v textu. Pokud se vzorek v textu nevyskytuje, funkce vrátí číslo -1.

```
#include <stdio.h>
#include <string.h>

int match(char pat[], char t[]) { /* DOPLŇTE */ }

int main (void) {
  char pattern[20], text[100]; int n;
  printf("text: "); fgets(text, 100, stdin);
  printf("pattern: "); scanf("%20s", pattern);
  n = match(pattern, text);
  if (n == -1) { printf("No occurrence\n"); }
  else { printf("%s", text);
    for (int i=0; i<n; i++) putchar(' ');
    printf("^\n");
  }
  return 0;
}
```

Hledání čísla vyhovujícího predikátu

Napište funkci, která v celočíselném poli najde pozici prvního prvku vyhovujícího danému predikátu. Pokud se číslo v textu nevyskytuje, vrátí se „pozice“ -1.

```
#include <stdio.h>
#include <stdbool.h>

int find(bool(*p)(int), int n, int a[]) {
  int k;
  for (k = 0; k<n && !(*p)(a[k]); k++);
  return k<n ? k : -1;
}

bool mod4gt5(int n) { return n % 4 == 0 && n > 5; }

int main(void) {
  int a[100] = {0,1,2,3,4,5,6,7,8,9};
  printf("n%%4==0 && n>5: %d\n", find(mod4gt5, 10, a));
  return 0;
}
```

Co znamená `"n%%4==0"` ve formátovém řetězci?

Napište definici predikátu sudosti a jeho použití v `main`.

Řazení vkládáním

Seřaďte číselnou posloupnost v poli. Funkce řadí *na místě*, tedy bude měnit pole, které je jejím řazeným parametrem.

```
#include <stdio.h>

void insertsort(int n, int a[]) {
                                        /*  DOPLŇTE  */
}

int main(void) {
  printf("pocet prvku: "); scanf("%d", &n);
  printf("posloupnost: ");
  for (int i=0; i<n; i++) scanf("%d", &a[i]);

  printf("puvodni:  ");
  for (int i=0; i<n; i++) printf(" %3d",a[i]); putchar('\n');

  insertsort(n, a);

  printf("serazena: ");
  for (int i=0; i<n; i++) printf(" %3d",a[i]); putchar('\n');
}
```