MUNI
FI

Cvičenie 03

Funkcie, modulárne programovanie, pamäť, ukazovatele

Funkcie

- Názov
- Typ návratovej hodnoty
- Typy a názvy parametrov
- Telo

```
int foo(int x, char c, long *ptr) {
  /* ... */
  return y;
}
```


MUNI FI

Funkcie

Deklarácia

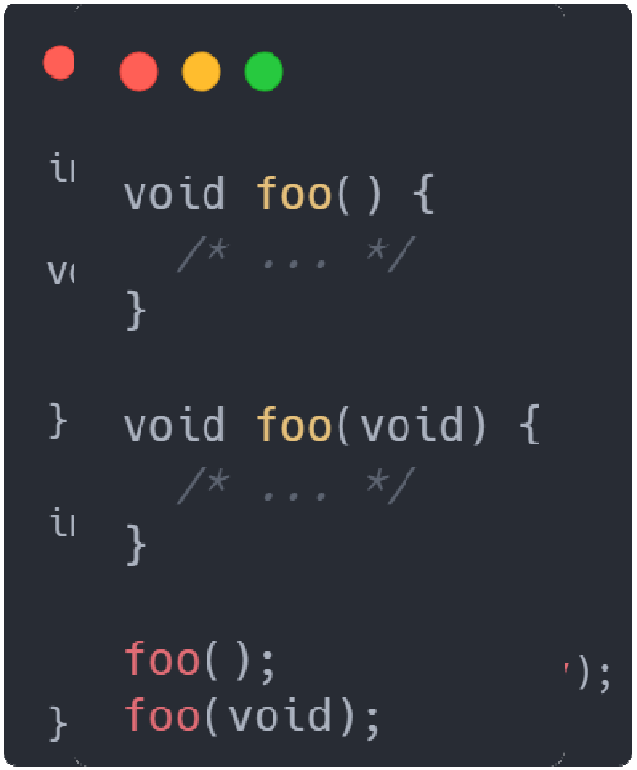
- „Prísľub“, že funkcia bude existovať
- Názov, typ návratovej hodnoty, **typy** parametrov
- Nutná pred použitím (volaním)

Definícia

- Chovanie funkcie
- Ako v deklarácii, ale aj s názvami parametrov + telo funkcie

Funkcie – premenné a parametre

- Počet parametrov:
  - Žiadne
  - Fixný počet
  - Premenný počet
- Rozsah
- Predávanie
  - Do funkcie
  - Von z funkcie



MUNI FI

Modulárne programovanie

- Hlavičkové súbory (.h)
  - Obsahujú deklarácie funkcií (a „typov“)
  - Použitie: `#include <filename.h>` – pre štandardné knižnice
    `#include „filename.h“` – pre užívateľské knižnice
    viac v štandarde jazyka C

- Implementačné súbory (.c)
  - Obsahujú definície funkcií
  - V súbore `lib.c` dobré použiť `#include „lib.h“`
  - Poskytujú sa prekladaču: `gcc -std=c99 ... -o a.out main.c lib.c`

Modulárne programovanie

```c
// main.c

#include "factorial.h"

int main(void) {
  int x = 0;
  if (scanf("%d", &x) != 1)
    return 1;

  int result = factorial(x);
  if (result == 0) {
    printf("Invalid input\n");
    return 1;
  }
  printf("%d\n", result);
  return 0;
}
```

```c
// factorial.h

#ifndef FACTORIAL_H
#define FACTORIAL_H

int factorial(int);
#endif
```

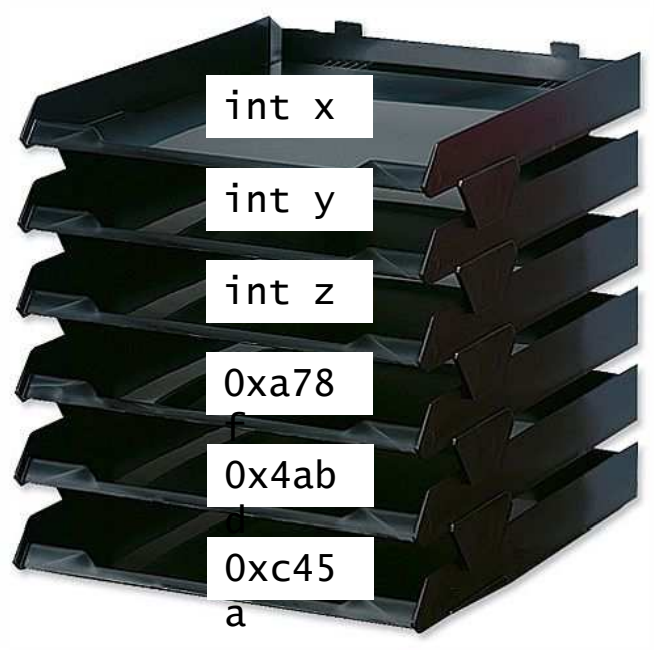
```c
// factorial.c

#include "factorial.h"

int factorial(int x) {
  if (x < 0)
    return 0;

  int result = 1;
  for (int i = 1; i < 0; i++) {
    result *= i;
  }

  return result;
}
```

```
gcc -std=c99 ... -o myFact main.c factorial.c
```

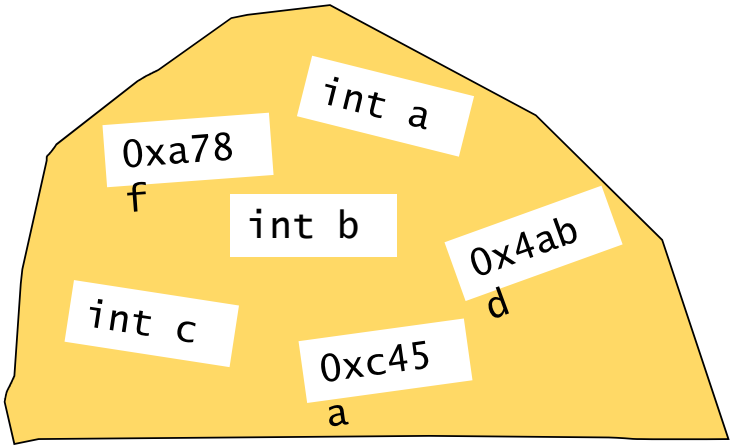

MUNI FI

Pamäť

Zásobník



Halda



MUNI FI

Ukazovatele

- Operátor &
  - Berie premennú
  - Vracia adresu premennej
- Typ ukazovateľ na premennú
  - Typ premennej
  - Označuje sa * za typom premennej, na ktorú ukazuje
- Ukazovateľ
  - Typ – ukazovateľ na typ premennej
  - Hodnota – adresa v pamäti
  - Získanie hodnoty na uloženej adrese pomocou operátora * - dereferencia
  - „Žiadna adresa“ - NULL

```c
int main(void) {
  int x = 0;
  int* pX = NULL;
  pX = &x;

  scanf("%d", pX);
  printf("%p: %d\n", pX, x);
  // 0x7ffc139a573c: 5

  printf("%p: %d\n", pX, *pX);
  // 0x7ffc139a573c: 5

  return 0;
}
```


MUNI FI

Funkcie a ukazovatele

```c
int divmod(int x, int y,
           int* quotient, int* remainder) {
  if (y == 0)
    return 0;

  *quotient = x / y;
  *remainder = x % y;

  return 1;
}

int divide(int* x, int y) {
  if (y == 0)
    return 0;

  *x /= y;
  return 1;
}
```


MUNI FI

Jednorozmerné statické polia

- Súvislá oblasť pamäte
- Homogénne – prvky rovnakého typu
- Prvky uchované zaradom
- Fixný počet prvkov (compile-time)

Jednorozmerné statické polia

- Veľkosť – počet prvkov * veľkosť prvku
- `typ_prvku názov_premennej[počet_prvkov];`
- Inicializácia – hodnota jedného alebo všetkých prvkov
- Prístup k prvkom pomocou `pole[index]`
- Bez informácie o počte prvkov

Jednorozmerné statické polia

```c
int main(void) {
  char array1[10];

  int size = 3;
  short array2[size] = { 0 };
  int array3[size] = { 1, 2, 3 };

  for (int i = 0; i < size; i++) {
    printf("%d ", array3[i]);
  }
  // 1 2 3
}
```


MUNI FI

Čo? Prečo? Ako?
???