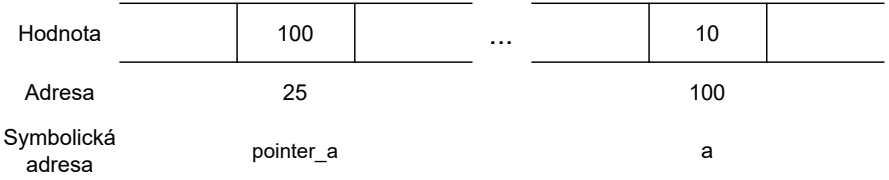
KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Práce s pamětí

Základy programování 2

Mgr. Markéta Trnečková, Ph.D.

Paměť



Ukazatele

deklarace:

```
typ *jmeno;
```

Nulový ukazatel: `NULL`

výpis:

```
printf("%p", ptr);
```

operátory:

- adresy: &
- dereference: *

Pointerová aritmetika

- **statické pole**: `int pole[] =` $\{1, 2, 3\}$
- pole je ukazatel na první prvek
- **konstantní pointer** - nelze mu přiřadit jinou hodnotu

Pointerová aritmetika

- K adrese lze přičíst nebo od ní odečíst nezáporné celé číslo
- Lze spočítat rozdíl adres stejného typu
- Adresy lze porovnávat stejně jako čísla

Pointerová aritmetika

Příklad

```
int pole[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

1. Jak zjistíte adresu 4. prvku pole?
2. Máme ukazatel, který ukazuje na některý prvek pole pole. Jak zjistíte index tohoto prvku v poli?
3. Máme dva ukazatele do stejného pole. Jak zjistíme, který z ukazatelů ukazuje na prvek, který je v poli dříve?

Příklad z minulého semináře

Příklad

Bez použití operátoru `sizeof()` zjistěte, jak velkou část paměti zabírají typy `char`, `int` a `double`.

Paměť

Správa paměti

- **Statická alokace**
- **Dynamická alokace**
  - Na zásobník
  - Na haldu

- deklarace proměnné
- definice proměnné
- rozsah platnosti identifikátoru

Funkce pro práci s pamětí

- **Knihovna**: `stdlib`
- **Přidělení paměti**: `malloc()`
- **Uvolnění paměti**: `free()`
- **Další funkce**:
  - `calloc()`
  - `realloc()`

Funkce pro práci s pamětí na haldě

Funkce `malloc()`

- Alokace paměti
- `void *malloc(size_t size)`
- `int *ptr_i = malloc(sizeof(int));`
- `ptr_i = (int *) malloc(sizeof(int));`

Funkce pro práci s pamětí na haldě

Funkce `calloc()`

- Alokace paměti + inicializace

```
ptr = calloc(n, sizeof(int));
ptr = (int *) malloc(n * sizeof(int));
```

Funkce pro práci s pamětí na haldě

Funkce `realloc()`

- Změna velikosti alokované paměti

```
void *realloc(void *adresa, size_t velikost);
```

Funkce pro práci s pamětí na haldě

- Pracujeme s ukazateli (ale i `[ ]`)
- Je však rozdíl mezi statickými a dynamickými poli
- **Příklad**: `alokace.c`

Příklad

Příklad (Celý kód: `pr2.c`)

```c
#include <stdio.h>
#include <stdlib.h>

int *data;
int velikost;
int hlava;

void init(int); // velikost pole data
void uvolni(); // uvolneni pameti alokovane pro data
void pridej(int); // pridani prvku

int main(){
    int i = 7;
    init(4);
    pridej(i);
    uvolni();
    return 0;
}
```

Cvičení

1. Zajistěte, aby se pole `data`, adekvátně zvětšovalo při přidávání prvků!
2. Doprogramujte k předchozímu příkladu funkci na vypsání prvků v poli data. Zavolejte tuto funkci po každém přidání prvku do pole.
3. V předchozím programu nahraďte použití globálních proměnných pomocí strukturovaného typu.
4. Doprogramujte k předchozímu příkladu funkci na odebrání posledního prvku v poli `data`.
5. Doprogramujte k předchozímu příkladu funkci, která zjistí zda se číslo předané jí jako argument nachází v datové struktuře.
6. Doprogramujte k předchozímu příkladu funkci, která smaže prvek předaný jí jako argument z datové struktury.
7. Upravte funkci mazání prvku tak, že pokud je počet prvků ve struktuře menší než polovina jeho velikosti, zmenší paměť alokovanou pro pole `data` na polovinu.