# Paměť

## Základy programování 2

Mgr. Markéta Trnečková, Ph.D.

Palacký University, Olomouc

# Paměť

- **alokovaná paměť**
- **dealokovaná paměť** = uvolněná paměť
- **2 druhy paměti**:
  - na zásobníku - alokace a dealokace automaticky
  - na haldě (heap) - o alokaci a dealokaci se stará programátor
- omezení paměti na zásobníku:
  - při alokaci je nutno znát velikost potřebné paměti
  - nelze vrátit pole jako návratovou hodnotu

# Paměť

## Example

```
// navratovy typ funkce: adresa typu int
int *vrat_pole(int velikost)
{
    /* Velikost neni konstanta! */
    int array[velikost];

    /* nejaky kod */

    /* chyba! array lezi v pameti, ktera bude po skonceni dealokovana. */
    return array;
}
```

# Paměť na heap

- `#include <stdlib.h>`
- **alokace:** `void *malloc(size_t velikost);`
- **přetypování:**
  - implicitní: přiřazení
  - explicitní: `(novy_typ) promenna;`
- **uvolnění paměti:** `void free(void *adresa);`

# Přetypování

## Example

```
int i = 5;
float f;

/* f = 1 */
f = i/4;

/* f = 1.25 */

f = ((float) i)/4;
```

# Paměť

## Example

```
// navratovy typ funkce: adresa typu int
int *vrat_pole(int velikost)
{
    int* array = malloc(sizeof(int)*velikost);

    /* nejaky kod */

    return array;
}
```

# Paměť na heap

- `malloc`
- `void *calloc(size_t polozky, size_t velikost);`
- **změna velikosti alokované paměti:** `void *realloc(void *adresa, size_t velikost);`

# Paměť

## Example

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

int *data;
int velikost;
int hlava;

void init(int);
void uvolni();
void pridej(int);

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

# Cvičení

1. Doprogramujte k předchozímu příkladu funkci na vypsání prvků v poli data. Zavolejte tuto funkci po každém přidání prvku do pole.
2. V předchozím programu nahraďte použití globálních proměnných pomocí strukturovaného typu.
3. Doprogramujte k předchozímu příkladu funkci na odebrání posledního prvku v poli data.
4. Doprogramujte k předchozímu příkladu funkci, která zjistí zda se číslo předané jí jako argument nachází v datové struktuře.
5. Doprogramujte k předchozímu příkladu funkci, která smaže prvek předaný jí jako argument z datové struktury.
6. 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.