Dynamisch geheugenbeheer in C

Andy Verkeyn
Vakgroep Informatie Technologie
Universiteit Gent

Versie: Augustus 2000
(Wijzigingen: 11/1998)

Overzicht

- **Header file: <stdlib.h>**

```
        void* malloc(size_t size);
        void free(void* memblock);
void* realloc(void* memblock, size_t size);
  void* calloc(size_t num, size_t size);
```

- **size_t**
  - type voor een grootte (bv. aantal bytes)
  - meestal een unsigned integer
- **size_t sizeof(type-name)**
  - retourneert de grootte van een type (in bytes)

malloc en free

- **Returnwaarde van m(emory) alloc(ation):**
  - void pointer naar het gealloceerde geheugen
  - NULL pointer indien niet genoeg geheugen beschikbaar
- **De teruggegeven void pointer moet naar het gewenste type gecast worden**
- **Steeds al het gealloceerd geheugen terug vrijgeven met free**

malloc en free

```
#include <stdlib.h>
#include <stdio.h>
void main() {
  char *string;
  string = (char*) malloc(80 * sizeof(char));
  if(string == NULL) /* NULL defined in stdlib.h */
    printf("Insufficient memory available\n");
  else {
    printf("Memory space allocated\n");
    free(string);
    printf("Memory freed\n");
  }
}
```

realloc

- Verandert de grootte (kleiner/groter) van een gealloceerd geheugenblok
- Kopiëert de inhoud naar een nieuwe locatie (indien nodig)

```
void function() {
  float *reals;
  /* allocate room for 20 floats */
  reals = (float*) malloc(20 * sizeof(float));
  /* reallocate room to hold 30 floats */
  reals = (float*) realloc(reals, 30 * sizeof(float));
}
```

calloc

- Alloceert een geheugenblok (≈ malloc)
- Initialiseert de inhoud op 0: c(lear) alloc(ate)

```
void function() {
  int i;
  float *reals;
  reals = (float*) calloc(20, sizeof(float));
  for(i = 0; i < 20; i++)
    printf("%f\n", *(reals + i));
}
```

- Vergelijk

```
malloc(20 * sizeof(float));
*(reals + i) ≡ reals[i]
```

sizeof

- **Laat toe het aantal elementen van een array te bepalen:**
  - sizeof(array): aantal bytes gealloceerd geheugen
  - sizeof(array[0]): aantal bytes van één array element

    sizeof(array)/sizeof(array[0]): aantal elementen

```
float p[] = {3.14, 6.28, 9.42, 12.56};
printf("\nElement grootte: %d", sizeof(p[0]));
printf("\nArray grootte: %d", sizeof(p));
printf("\nAantal elementen: %d",
  sizeof(p)/sizeof(p[0]));
```