Základy programovania 25ZS
Dynamická alokácia pamäte

Prednášajúci: prof. Gabriel Juhás
Cvičiaci: Milan Mladoniczky

Dynamická alokácia pamäte

Pri práci s poliami a štruktúrami niekedy nie je známa ich veľkosť v čase definície.

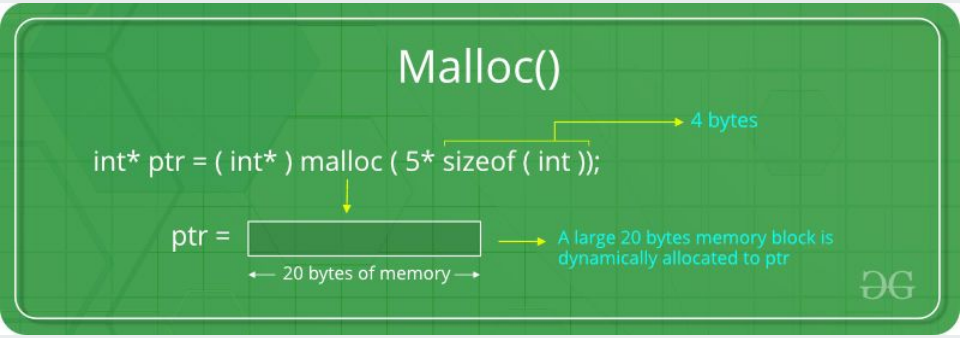
Pre tieto potreby umožňuje jazyk C tzv. dynamickú alokaciu pamäte. T.j. vymedziť časť pamäte určitej veľkosť počas behu programu.

Funkcie alokácie vždy vrátia pointer typu void* a developer je povinný pointer pretypovať na požadovaný typ.

- **malloc** - alokácia novej pamäte špecifikovanej veľkosti
- **calloc** - alokácia definovaného počtu blokov pamäte špecifikovanej veľkosti
- **realloc** - zmena veľkosti alokovanej pamäte
- **free** - uvoľnenie alokovanej pamäte

malloc

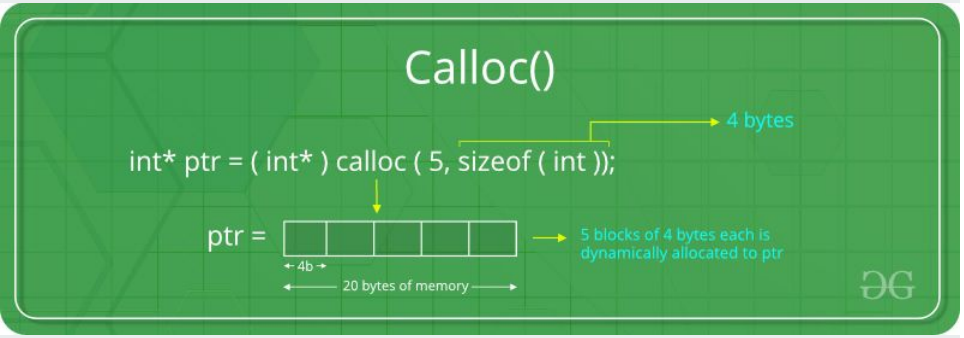
memory allocation




```
int* ptr;
int n = 5;

ptr = (int*)malloc(n * sizeof(int));

if(ptr == NULL) {
     printf("Memory failed to allocate");
     return 1;
}
for (i = 0; i < n; ++i) {
     printf("%d, ", ptr[i]);
}
```

calloc

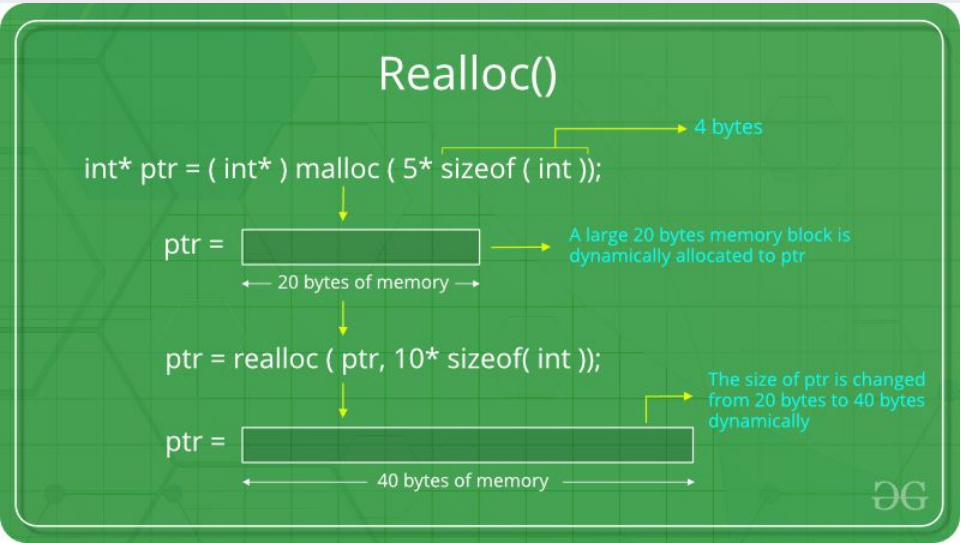
contiguous allocation




```
int* ptr;
int n = 5;

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

if(ptr == NULL) {
     printf("Memory failed to allocate");
     return 1;
}
for (i = 0; i < n; ++i) {
     printf("%d, ", ptr[i]);
}
```

realloc

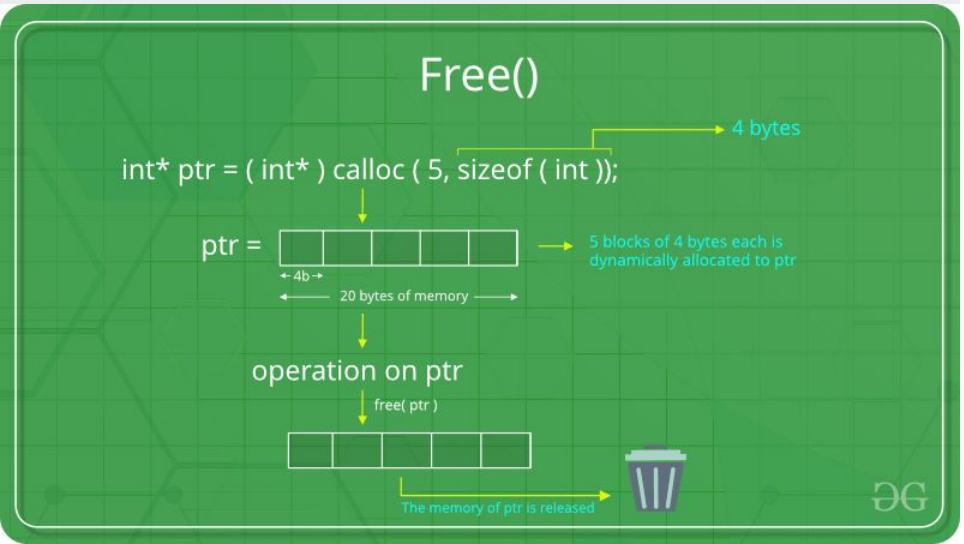
memory re-allocation




```
int* ptr;
int n = 5;

ptr = (int*)calloc(n, sizeof(int));
if(ptr == NULL) {
     printf("Memory failed to allocate");
     return 1;
}
for (i = 0; i < n; ++i) {
     printf("%d, ", ptr[i]);
}

n = 10;
ptr = (int*)realloc(ptr, n * sizeof(int));

for (i = 0; i < n; ++i) {
     printf("%d, ", ptr[i]);
}
```

free

memory de-allocation




```
int* ptr;
int n = 5;

ptr = (int*)malloc(n * sizeof(int));

if(ptr == NULL) {
     printf("Memory failed to allocate");
     return 1;
}
printf("Allocated: %p", ptr);

free(ptr);

printf("Memory cleared: %p", ptr);
```

Praktická ukážka –>

https://zapr.interes.group/exercises/exercise-7