

Programmation avancée

Allocation Dynamique

Walter Rudametkin

Walter.Rudametkin@polytech-lille.fr
<https://rudametw.github.io/teaching/>

Bureau F011
Polytech'Lille

8 février 2016

1 / 14

Allocation de mémoire

Variables automatiques

- Var de bloc, paramètres
- Créées automatiquement à l'exécution
- Allocation dynamique sur la pile

Variables dynamiques

- Créées et détruites dynamiquement et explicitement
- Fonctions malloc et free
- Allocation sur le tas (heap)

2 / 14

Erreur d'allocation

```
/* À ne pas faire */  
  
int * allouer_entier() {  
    int var_static ; //allocated on the stack  
    printf("var_static address is : %p\n",  
          &var_static);  
  
    return &var_static ;  
}
```

3 / 14

Allocation dynamique — malloc

Fonction malloc

- void * malloc (int taille);
 - Alloue dynamiquement dans le tas un espace de taille octets
 - Résultat : *pointeur non typé* vers la zone allouée
 - Pointeur peut être converti automatiquement vers le type désiré (conversion implicite)
 - Besoin de `#include<stdlib.h>`

4 / 14

Allocation dynamique — Exemples

Allocation dynamique d'un entier

```
int *pt;  
pt = (int *) malloc(sizeof(int));  
*pt = 42; //utilisation
```

Allocation dynamique d'un tableau d'entiers

```
int n; int *pt;  
scanf("%d", &n);  
pt = (int *) malloc(n*sizeof(int));  
//Accès  
*pt = 11; //premier élément  
*(pt+1) = 22 ; //deuxième  
pt[2] = 33 ; //troisième  
*(pt+n-1) = 9876 ; //dernier
```

5 / 14

Allocation dynamique — Structures

```
1 #include <stdio.h>  
2 #include <stdlib.h>  
3  
4 typedef struct {  
5     int j,m,a;  
6 } Date;  
7  
8 int main(){  
9     /*Date *pDate = (Date *) malloc(sizeof(Date));*/  
10    Date *pDate = malloc(sizeof *pDate);  
11  
12    printf("sizeof Date:%lu | sizeof *pDate:%lu\n",  
13          sizeof(Date), sizeof *pDate);  
14  
15    /*ex. utilisation :*/  
16    scanf("%d%d%d",&(pDate->j),  
17          &(pDate->m),  
18          &(pDate->a));  
19  
20    printf("Date %d/%d/%d\n", pDate->j,  
21          pDate->m,  
22          pDate->a);  
23    free(pDate); pDate = NULL;  
24 }
```

6 / 14

Allocation dynamique — Structures

Tableau de structures

```
int n;
Date *pt;
scanf("%d", &n);
/*pt = (Date *) malloc( n * sizeof(Date));*/
pt = malloc( n * sizeof *pt);

/*utilisation: notation equivalent*/
scanf("%d%d%d", &(pt[0].j),
      &((pt+0).m),
      &pt[0].a);
printf("Date %d/%d/%d\n", pt[0].j, pt[0].m, pt[0].a);
free(pt); pt = NULL;
```

7/14

Allocation dynamique — Liste contiguë

```
typedef Date * PtDate;

typedef struct {
    PtDate * espace; //vecteur de PtDate alloué dynamiquement
    int dernier;
} Liste;

int n; Liste l;
l.dernier = -1;

scanf("%d", &n); //nb de pointeurs à Date

l.espace = malloc (n * sizeof *l.espace);
//Alternative
l.espace = (PtDate *) malloc (n * sizeof (PtDate));
```

8/14

Allocation dynamique — Liste contiguë

```
printf("\nAllocate:\n");
for(int i=0 ; i<n ; i++){
    l.dernier+=1;
    l.espace[l.dernier] = malloc(sizeof **l.espace);
    /*l.espace[l.dernier] = (PtDate) malloc(sizeof(Date));*/
    l.espace[l.dernier]->j=i;
    l.espace[l.dernier]->m=i;
    l.espace[l.dernier]->a=i;
}
printf("\nDernier date: %d\n", l.dernier);
for(int i=0 ; i<=l.dernier ; i++){
    printf("Date[%d] %d/%d/%d\n", i, l.espace[i]->j,
                                              l.espace[i]->m,
                                              l.espace[i]->a);
}
```

9/14

Fonction free

- ▶ void free(void *ptr);
 - ▶ libère l'espace mémoire pointé par ptr (précédemment alloué)
- ▶ Exemple d'utilisation:
Suppression du dernier élément de la liste

```
free(l.espace[l.dernier]);
l.dernier -= 1;
```

10/14

Listes chaînées — Implantation en C

```
//Définition
typedef struct cellule {
    int valeur;
    struct cellule *suivant;
} Cellule;

typedef Cellule *Liste, *Ptcellule;
```

```
/* liste vide */
Liste l; l = NULL;
```

```
/* accès aux champs */
Ptcellule p ; //N'oubliez pas de l'initialiser
//...
(*p).valeur; /* ou */ p->valeur ;
(*p).suivant; /* ou */ p->suivant ;
```

11/14

Listes chaînées — Implantation en C

```
//Fonction qui alloue une cellule en mémoire
Ptcellule allouer(){
    return( (Ptcellule) malloc(sizeof(Cellule)) ) ;
}
```

```
//Allocation d'une cellule
Ptcellule p = allouer();
```

```
//Libération d'une cellule
free(p);
```

12/14

Listes chaînées — Recherche d'un élément

//Exemple

```
int recherche(int x, Liste l) {
    int existe ; Pcellule p;
    p = l;
    while ( (p != NULL) && (p->valeur != x) ) {
        p = p->suivant;
    }
    existe = (p!=NULL);
    return (existe);
}
```

13/14

Listes chaînées — Exemple: ajout en tête

```
void ajout_tete (int x, Liste *pL){
    /* pL en D/R */
    Pcellule p;
    p = allouer();
    p->valeur = x;
    p->suivant = *pL;
    *pL = p;
}

int main(){
    int x ; Liste l=NULL;
    scanf("%d", &x);
    while (x > 0) {
        ajout_tete(x, &l);
        scanf("%d", &x);
    }
}
```

14/14