Allocations dynamiques

Avec l'utilisation des pointeurs il souvent nécessaire d'allouer des espaces mémoires d'une taille spécifiques (pour des structures, des tableaux, etc…).
Une allocation peut être faite à l'aide de diverses fonctions (realloc, calloc), le cas le plus courant est l'utilisation de « malloc », syntaxe (pour un tableau par exemple) :

```
malloc(nombreDeDonnees * sizeof(typeDeDonnée));
```

Renvoie la zone allouée ou NULL si l'allocation est impossible.
Quand on alloue la mémoire il ne faut pas oublier de la libérer dès qu'elle n'est plus nécessaire (mais pas avant) :

```
free(pointeur);
```

Exemples de réécriture de fonctions standards qui utilisent des pointeurs et des allocations mémoires :

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

int myStrlen (char *s){
    int longu=0;
    while(*s!='\0'){ //Equivaut à while(*s)
        longu++;
        s++;
    }
    return longu;
}

char* myStrCopy (char *desti, char *src){
    int decalage = 0;
    while(*src){
        *desti=*src;
        desti++;
        src++;
        decalage++;
    }
    return desti-decalage;
}

char* myStrDup (char *src){
    char* tempo = malloc(myStrlen(src) * sizeof(char)+1); //+1 pour '\0'
    return myStrCopy(tempo, src);
}

char* myStrCherch (char *s, char c){
    while(*s!='\0' && *s!=c)
        s++;
    return (*s!='\0') ? s : NULL;
}
```

```c
char* myStrTok (char *src, char *delim){
    char* delimDep=delim;
    static char* s; static char* lastSrc;

    if(src!=NULL){
        s=src;
        while(*src)
            src++;
        lastSrc=src+1;
    }
    char* sDep=s;

    while(*s){
        while(*delim){
            if(*s == *delim){
                break;
           }
            delim++;
        }
        if(*s == *delim){
            break;
        }
        delim = delimDep;
        s++;
    }
    s++;
    if(lastSrc<s)
        return NULL;
    return sDep;
}

int main(){
    char* chain1="Nous c'est le goût !";
    printf("Longeure de chain1 : %d\n",myStrlen(chain1));
    char* chain2="Tada";
    printf("Longeure de chain2 : %d\n",myStrlen(chain2));
    char* chainDes = malloc(myStrlen(chain1) * sizeof(char)+1);
    printf("chainDes : %s\n",myStrCopy(chainDes,chain2));
    printf("chainDes : %s\n",myStrDup(chain2));
    printf("Recherche a dans Tada : %s\n",myStrCherch(chain2, 'a'));
    printf("Recherche x dans Tada : %s\n",myStrCherch(chain2, 'x'));
    printf("myStrTok 1 : %s\n",myStrTok(chain1, " "));
    //Renvoie : Nous c’est le goût !
    printf("myStrTok 2 : %s\n",myStrTok(NULL, " ")); //c’est le goût !
    printf("myStrTok 3 : %s\n",myStrTok(NULL, " ")); //le goût !
    printf("myStrTok 4 : %s\n",myStrTok(NULL, " ")); //goût !
    printf("myStrTok 5 : %s\n",myStrTok(NULL, " ")); //!
    printf("myStrTok 6 : %s\n",myStrTok(NULL, " ")); //NULL
    char* chainEtrange="Nous c'est l\te goût !";
    printf("myStrTok 1 : %s\n",myStrTok(chainEtrange, " \t"));
    //Renvoie : Nous c’est l    e goût !
    printf("myStrTok 2 : %s\n",myStrTok(NULL, " \t")); //c’est l    e goût !
    printf("myStrTok 3 : %s\n",myStrTok(NULL, " \t")); //l    e goût !
    printf("myStrTok 4 : %s\n",myStrTok(NULL, " \t")); //e goût !
    printf("myStrTok 5 : %s\n",myStrTok(NULL, " \t")); //gout !
    printf("myStrTok 6 : %s\n",myStrTok(NULL, " \t")); //!

    return 0;
}
```