

Thibault Schowingz

NE RIEN INSCRIRE SUR LA DONNEE

`s = new(nothrow) char[n+1];`

Problème

```
if (s != NULL) { 1) Pro
    for (int i = 0; i < n; i++) ela
        s[i] = c; cha
    s[n] = '\0'
    } Sé
    return s; a a
```

Sémantique : Livre une chaîne de caractères comprenant n fois le caractère c , s'il y a assez de mémoire pour réaliser l'opération, sinon renvoie un pointeur nul.

A diagram showing a pointer variable '3' in a box, with an arrow pointing to the first element 'A' in an array. The array is represented as a row of four boxes containing 'A', 'B', 'C', and 'D'.

$$ptr \rightarrow \dots ptr$$

*ptr -- → *ptr

A	B	C	10
---	---	---	----

 pta devrait
se trouver sur C

Problème sur le positionnement

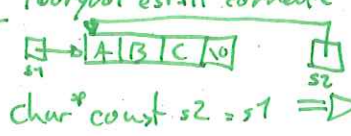
১৮

L7: $\text{ptr} \rightarrow \text{ptr}$

L8: $\ast_{pt_r} \dashv \dashv \ast_{pt_r}$

- La ligne no 1 ne doit en aucun cas être modifiée
- Aucune ligne de code ne doit être ajoutée ou supprimée
- Aucune fonction prédéfinie de la librairie standard ne doit être utilisée

3) En supposant la fonction *inverser* précédente dûment corrigée, on considère maintenant les extraits de code (indépendants les uns des autres) suivants :

- constant*
La chaîne est constante, on ne peut pas la modifier.
- ✗ a) `inverser((char*) "ABC");`
- ✓ b) `char s[] = "ABC";
inverser(s);`
- ✓ c) `char s1[] = "ABC";
char* s2 = s1;
inverser(s2);`
- ✓ d) `char s1[] = "ABC";
char* const s2 = s1;
inverser(s2);`
- Pourquoi est-il correcte*

char const s2 = s1 => ptr constant, pas const char*

Indiquez lequel (lesquels) de ces extraits n'est (ne sont) pas correct(s) et pourquoi.

- 4) Proposez une implémentation de la fonction *adrDernierCaract* dont la sémantique est la suivante :
adrDernierCaract est une fonction sans valeur de retour qui prend comme unique paramètre d'entrée une chaîne de caractères "à la C" et qui renvoie en paramètre de sortie l'adresse du dernier caractère de cette chaîne (*NULL* si la chaîne est vide).

IMPORTANT

- La fonction doit être écrite (paramètres inclus) en utilisant **exclusivement le formalisme pointeur**
- Aucune variable locale ne doit être déclarée dans la fonction
- Aucune fonction prédéfinie de la librairie standard ne doit être utilisée

```
void adrDernierCaract(const char* ch, [const] char** adr) {
    if (ch == NULL || *ch == '\0') {
        *adr = NULL;
        return;
    }
    while(*ch != '\0') ch++;
    *adr = (char*) --ch;
}

// ou if (!ch || !*ch)
// ou if (!(ch && *ch))

// ou while (*ch) ch++;
// ou while (*++ch), car pas de chaîne vide possible.
(char*) inutile si : const char** adr
```

const type → type**

5) La fonction C++ *rtrim* (pour *right trim*) ci-dessous est censée permettre la suppression de tous les éventuels caractères espace-blanc¹ présents à la fin de la chaîne *s*.

¹ Les caractères espace-blanc (*white-space characters* en anglais) sont :

' ', '\t', '\n', '\v', '\f' et '\r'

```

1 void rtrim(char* s) {
2     if (s) {
3         size_t i = strlen(s);
4         while (s[i] == ' ' || s[i] == '\t' || s[i] == '\n' || s[i] == '\v' || s[i] == '\f' || s[i] == '\r')
5             i--;
6         memcpy(s, s, i);
7         s[i] = '\0';
8     }
9 }
10

```

*if (s && *s)*
long long i = strlen(s) - 1
isspace <ctype>
chevauchement

Cette fonction contient toutefois des maladroites et/ou des erreurs.

Indiquez le no des lignes problématiques et proposez dans chaque cas un correctif.

IMPORTANT

- La ligne no 1 ne doit en aucun cas être modifiée
- Aucune ligne de code ne doit être ajoutée ou supprimée

```

void rtrim(char* s) {
    if (s && *s) {
        long long i = strlen(s) - 1;
        while (i >= 0 && isspace(s[i]))
            i--;
        memmove(s, s, i + 1); // NB : ligne inutile
        s[i + 1] = '\0';
    }
}

```

garanti qu'on reste dans le tableau (si [] le isspace(s[-1]) ne doit pas être fait)
remplacé par '\0'

[-A | B | L | \0]

Problème 2 (1.5 point)

Soit le code (incomplet) suivant :

<à compléter 1>

```
int main() {  
  
    CoteAmour coteAmour;  
    const unsigned I_MAX = 7;  
  
    coteAmour = CoteAmour::UN_PEU;  
    cout << "J'aime C++" << endl;  
    for (unsigned i = 0; i < I_MAX; ++i)  
        cout << coteAmour + i << "...";  
    cout << endl << endl;  
  
    coteAmour = CoteAmour::UN_PEU;  
    cout << "J'aime C++" << endl;  
    for (unsigned i = 0; i < I_MAX; ++i)  
        cout << coteAmour++ << "...";  
    cout << endl << endl;  
  
    coteAmour = CoteAmour::PAS_DU_TOUT;  
    cout << "J'aime C++" << endl;  
    for (unsigned i = 0; i < I_MAX; ++i)  
        cout << ++coteAmour << "...";  
    cout << endl << endl;  
  
    system("PAUSE");  
    return EXIT_SUCCESS;  
}
```

<à compléter 2>

Complétez, le plus proprement possible, les parties <à compléter> du code ci-dessus, de telle sorte que, à l'exécution, celui-ci affiche trois fois de suite :

```
J'aime C++  
un peu...beaucoup...passionnement...a la folie...pas du tout...un peu...beaucoup...
```

IMPORTANT

- Le code de la fonction *main* ne doit en aucun cas être modifié
- Implémentez le type *CoteAmour* en tant que **type énuméré fortement typé**

Problème 3 (1.5 point)

- 1) Concevoir, de la manière la plus propre et la plus évolutive possible, **la spécification d'une librairie** permettant de modéliser des vaisseaux spatiaux conformément au cahier des charges suivant :
 - Un vaisseau spatial a un nom, est composé d'un équipage pouvant comporter zéro, un ou plusieurs membres (dont seul le nom, pour l'instant, nous intéresse) et est soit un vaisseau de combat, soit un vaisseau d'exploration;
 - Si le vaisseau spatial est un vaisseau de combat, on souhaite enregistrer, en plus de son nom, son mode de propulsion (type énuméré dont les valeurs possibles sont : standard, supraluminique, ionique) et s'il est équipé ou non de canons laser;
 - Si le vaisseau spatial est un vaisseau d'exploration, on souhaite enregistrer, en plus de son nom, son rayon d'action (nombre réel exprimé en milliards de km)
- IMPORTANT**
 - Pour les chaînes de caractères, considérer le type *string*
 - Il n'est PAS demandé de proposer quelque fonction que ce soit (constructeur, fonction membre ou non) dans la spécification demandée
- 2) En supposant le point 1) résolu et que le compilateur utilisé supporte la norme C++ 2011, déclarer, **de la manière la plus concise possible**, les 2 vaisseaux spatiaux suivants :
 - Starfighter : un vaisseau de combat à propulsion ionique, équipé de canons laser et ayant pour équipage Joe et Jack
 - X-Wing : un vaisseau d'exploration sans équipage dont le rayon d'action est de 63.2 milliards de km

Problème 1 (0.65)

Schowing Thibault

3.6

1) 0.4 char* strcpy (char c, unsigned int n)

{
char* zone;

char

zone = new(nothrow) [(n+1) * sizeof(char)];
s = new(nothrow) char[n+1]

confusion avec malloc/calloc!

if (zone)

{

for (register unsigned int i=0; i<n; i++)

{

zone[i] = c;

} (char) zone[n] = '\0';

} return (char*) zone;

else

ne sert à rien

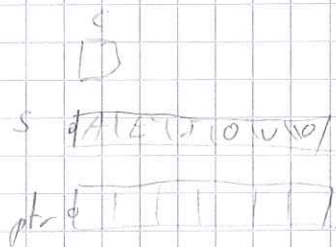
return zone;

// si est null.

→ possible message d'erreur
mémoire insuffisante

2) fin à l'unité?

while (s < ptr) ne s'exécute jamais car ptr = s
à la 1ère itération du for
→ while (s <= ptr)



L. 1 *s++ → *(s++) = ptr;

L. 8 *ptr-- → *(ptr--) = c;

! *s++ ≡ *(s++)

3)

0.1 a) `(char*) "ABC"` → constant ✓

d) ambiguë

sz pointe sur une chaîne constante

or il pointe sur sz qui ne l'est pas

`inverser(sz)` ne fonctionnera pas car sz → constant

Faux!

4)

void `adr DernierCaract` (`char* s`, `**char & adr`)

```
{
    while(*s && s++) {
        *adr = s;
    }
}
```

Si s vaut NULL initialement
→ crash!

5)



```
size_t i = strlen(s) - 1;
memcpy(s, s, i+1);
```

0.15

Travail avec les indices par la suite et on copie le '0' au premier passage

Schowing Thibault

0	1	2	3	4	5
h	e	e	e	o	o

strlen = 5
L'ensemble passe le '0'

h	e	e	e	o	o
0	1	2	3	4	5
4	3				

Problème 2 0.8

```
#include <iostream>
#include <cstdlib>
// marque #include <string>
```

Par rapport au tableau

tab
 $\frac{\text{sizeof(Cotes_Amour)}}{\text{sizeof(string)}}$

```
using namespace std;
const unsigned int NB_ELEM = 5; // utiliser sizeof!
enum class CotesAmour { UN_PEU, BEAUCOUP, PASSIONNEMENT, A_LA_FOLIE, PAS_DOUTOUT };
ostream & operator << (ostream & os, const CotesAmour & c);
CotesAmour operator + (CotesAmour & c, int i);
CotesAmour operator ++ (CotesAmour & c); // pré-inc.
CotesAmour operator ++ (CotesAmour & c, int n);
```

```
string tab[] = { "un peu", "beaucoup", "passionnement", "à la folie", "pas du tout" };
```

```
ostream & operator << (ostream & os, const CotesAmour & c) {
    // faux
    // cout << tab [(int)c] % NB_ELEM;
    return os << tab[(int)c];
    return os;
}
```

```
CotesAmour operator + (CotesAmour & c, int i) {
    return CotesAmour((int)c + i) % NB_ELEM;
}
```

```
CotesAmour operator ++ (CotesAmour & c) {
    return CotesAmour((int)c + 1);
    // faux
    // return c = c + 1;
}
```

```
CotesAmour operator ++ (CotesAmour & c, int i) {
    CotesAmour tmp = c;
    // tmp = c + i;
    c = c + i;
    return tmp;
}
```

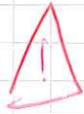
la cyclicité doit être prise en compte dans les opér. + et ++ et pas dans l'opér <<

0.15

Problème 3 (0.6)
 1) typedefs vector<string> Equipage; on peut faire plus évolutif
 Enum prop { standard, supraluminaire, ionique };
~~Enum type { combat, exploration }; faux~~



marque #ifndef...
 #include



les mots réservés
 s'écrivent entièrement
 en minuscules !!! oups

combat Expl
 ↳ propE ↳ rayon
 ↳ canon V/N ~~uniquement~~

~~struct~~ explo {
 unsigned float rayon;
 };
 ??? pas de sens!

enum typeVais. { COMBAT, EXPLO };
 enum ModeProp { STANDARD, sup... };

~~struct~~ combat {
 prop propulsion;
 bool canon;
 };

struct V.Combat {
 ModeProp mode;
 bool canon; }
 union specificite {
 VaisseauCombat
 VaisseauExplo

OK

struct MembreEquipage
 { string Nom; }

typedef vector<MembreEquipage

struct Vaisseau {
 string nom;
 Equipage equipage;
 Union categorie {
 explo;
 combat;
 };
 vaisseau1 =

! marque champ discriminant!

Doublé
 faux!!

2) VaisseauV { "Starfighter", { "Joe", "Jack" } };
 a) vaisseau1.categorie.combat.prop = ionique;
 " " " " canon = true;
 Vaisseau vaisseau2 = { "X-Wing", { }, { 62, 3 } }
 b)

7.5 pts