

Travail écrit no. 1

(Durée : 2 périodes)

Thibault Schowing

NE RIEN INSCRIRE SUR LA DONNEE

Problème 1 (2 points)

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

1) Proposez une implémentation C++ de la fonction *strelab* (abréviation de *string elaboration*) dont le prototype et la sémantique sont donnés ci-dessous :

for (uint i = 0; i < n; i++)

s[i] = c;

s[n] = '\0';

return s;

char* strelab(char c, unsigned int n);

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

2) La fonction ci-dessous est censée inverser la chaîne de caractères *s* passée en paramètre :



correctif

```
1 void inverser(char* s) {  
2     if (s) {  
3         char c, *ptr;  
4         for (ptr = s; *ptr; ptr++)  
5             while (s < ptr) {  
6                 c = *s;  
7                 *s++ = *ptr;  
8                 *ptr-- = c;  
9             }  
10    }  
11 }
```

ptr --> --ptr

**ptr-- --> *ptr*



Problème sur le positionnement

Cette fonction n'est toutefois pas correcte.

Indiquez le no de la ligne (des lignes) fautive(s) et proposez un (des) correctifs.

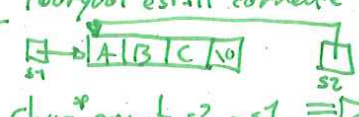
*L7: *ptr --> *ptr*

*L8: *ptr --> *ptr*

IMPORTANT

- 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.
- X 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') { // ou if (!ch || !*ch)
        *adr = NULL; // ou if (!(ch && *ch))
        return;
    }
    while(*ch != '\0') ch++; // ou while(*ch) ch++;
    *adr = (char*)--ch; // ou while(*++ch), car pas de chaîne vide possible.
} // (char*) inutile si : const char** adr
```

const type → type**

- ¹ Les caractères espace-blanc (*white-space characters* en anglais) sont : ' ', '\t', '\n', '\v', '\f' et '\r'.

Cette fonction contient toutefois des maladroites et/ou des erreurs.
Indiquez le no des lignes problématiques et proposez dans chaque cas un correctif.

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

Hes·SO
Haute Ecole Spécialisée
de Suisse occidentale

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)

(3.6)

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

{
char* zone;char
s = 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++)

{
(char) zone[i] = c;} (char) zone[n] = '\0';
return (char*) zone;

else

ne sert à rien

return zone;

// s'il est null. → possible message d'erreur
→ ou null ptr

if (s != null) {

for (uint i=0; i<n;i++)

s[i] = c;

s[n] = '\0';

} return s;

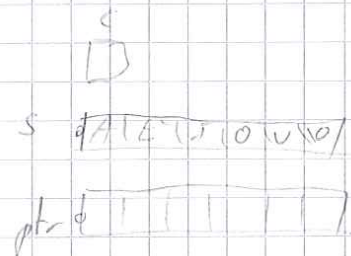
2) tri à bulle?

while (s < ptr) ne s'opère jamais car ptr = s
à la 1^{re} 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(~~const~~ ✓ `char* s`, `** char & adr`) ^{???}

{

while(`*s` ~~signifiant 0~~ `&& s++`) ^{???}

`*adr = s`; ^{???}

}

Si s vaut NULL initialement
→ crash!

Schoning Thibault

0	1	2	3	4	5
L	E	L	L	O	0

strlen = 5

L'exemple parle d'

L	E	L	L	O	0
---	---	---	---	---	---

0	1	2	3	4	5
---	---	---	---	---	---

4	3
---	---

pour on travaille avec les indices par la suite et on compte le '0' au premier passage

Problème 2 0.8

#include <iostream>

#include <cstdlib>

ma-pse #include <string>

using namespace std;

const unsigned int NB_ELEM = 5;

utiliser sizeof!

enum class CoteAmour { UN_PEU, BEAUCOUP, PASSIONNEMENT, A_LA_FOLIE, PAS_DOUTOUT };

ostream & operator << (ostream & os, const CoteAmour & c);

CoteAmour operator + (CoteAmour & c, int i);

CoteAmour operator ++ (CoteAmour & c); // pré-inc.

CoteAmour operator ++ (CoteAmour & c, int n);

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

ostream & operator << (ostream & os, const CoteAmour & c)

{
const << tab[(int)c % NB_ELEM];

return os << tab[(int)c];

return os; }

CoteAmour operator + (CoteAmour & c, int i) {
return CoteAmour((int)c + i % NB_ELEM);

CoteAmour operator ++ (CoteAmour & c) { // pré

return CoteAmour((int)c + 1); }

CoteAmour operator ++ (CoteAmour & c, int i) { // post

CoteAmour tmp = c;

tmp = c + i;

c = c + i; return tmp; }

Par rapport au tableau

sizeof(CoteAmour) / sizeof(string)

tab

Non! unsigned!

0.1

0.15

0.1

0

0.1

0.15

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

Problème 3 (0.6)
 1) types des vecteur < 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 X/N insignifiant

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

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

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

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

struct MembreEQUIPAGE
 { string Nom; }
 typedef vector<MembreEQUIPAGE

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

marque champ discriminant!
 Darkement
 faux!!

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

7.5 pts