Sujet 1 : Références	
Buts :	prototype, passage par référence et référence constante, objet *string*.

Note : il est demandé de répondre dans l'ordre aux questions posées !

Exercice 1 : Calculatrice - types natifs

Étape 1

On souhaiterait disposer d'une fonction `saisieDonnees` permettant de demander à l'utilisateur les deux opérandes nécessaires pour effectuer une opération binaire simple ainsi que l'opérateur qu'il souhaite utiliser (caractère). On se limitera ici au cas où les deux opérandes sont des entiers et aux opérateurs +, -, *, / et %. Cette fonction retournera `true` si l'utilisateur a souhaité saisir les données, `false` sinon.

Q1. Quel est le prototype de la fonction `saisieDonnees` ?

Q2. Écrivez la fonction `main` permettant d'appeler la fonction `saisieDonnees`. Vous veillerez à afficher les données saisies après l'appel de cette fonction. Par exemple, si l'utilisateur a saisi 4, 6 et /, le programme doit afficher :

```
Opération demandée : 4 / 6
```

Q3. Donnez la définition de la fonction `saisieDonnees` dont l'appel pourra produire la trace d'exécution suivante :

```
Souhaitez vous saisir des données ? (0-non, 1-oui) 5
Souhaitez vous saisir des données ? (0-non, 1-oui) 1
Entrez l'opérande 1 (entier) : 4
Entrez l'opérande 2 (entier) : 6
Entrez l'opérateur (+, -, *, / ou %) : T
Entrez l'opérateur (+, -, *, / ou %) : /
```

Note : on considère que l'utilisateur respecte toujours le type des données demandées à la saisie, il ne saisit pas une chaîne de caractères quand un entier est attendu, etc.

Étape 2

On souhaiterait disposer d'une fonction `calcul` permettant d'effectuer une opération binaire simple connaissant la valeur des deux opérandes (entiers) ainsi que le symbole correspondant à l'opération (caractère +, -, *, / ou %), ces informations étant passées en paramètre. Cette fonction retournera le résultat sous forme d'un entier et aura un paramètre supplémentaire permettant de savoir si l'opération a pu être effectuée (par exemple si l'utilisateur demande une division par 0, l'opération est impossible). Après l'appel de la fonction, ce paramètre aura pour valeur `true` si le calcul a pu être effectué, `false` sinon.

Voici une trace d'exécution correspondant au déroulement du programme :

```
Souhaitez vous saisir des données ? (0-non, 1-oui) 1
Entrez l'opérande 1 (entier) : 5
Entrez l'opérande 2 (entier) : 3
```

```
Entrez l'opérateur (+, -, *, / ou %) : -
5-3=2
Souhaitez vous saisir des données ? (0-non, 1-oui) 1
Entrez l'opérande 1 (entier) : 6
Entrez l'opérande 2 (entier) : 0
Entrez l'opérateur (+, -, *, / ou %) : /
Opération impossible
Souhaitez vous saisir des données ? (0-non, 1-oui) 0
```

Q4. Donnez la définition complète de la fonction `calcul`.

Q5. Complétez la fonction `main` afin d'obtenir une exécution cohérente avec la trace ci-dessus.

Exercice 2 : L'objet *string*

Note : une fiche concernant la manipulation de l'objet `string` est fournie à la fin de ce sujet. **Vous veillerez à ne pas parcourir les chaînes avec des boucles,** mais privilégierez l'utilisation des fonctions fournies en annexes.

Question 1

Écrivez une fonction `extrait` qui prend en paramètres une chaîne de caractères `chaine`, un entier `position` et un caractère `terminaison`. Cette fonction recherche dans la `chaine`, à partir de `position`, la première occurrence du caractère `terminaison`. Elle retourne une sous-chaîne construite avec les caractères de `chaine` compris entre `position` et l'occurrence de `terminaison` trouvée. Si le caractère `terminaison` n'a pas été trouvé, la sous-chaîne sera construite avec tous les caractères depuis `position` jusqu'à la fin de la `chaine`.
Attention : la fonction manipulera la chaîne de caractères `chaine` d'origine et non sa copie. Cependant il faudra la protéger de toute modification.

```
extrait( "Quelle surprise", 7, 'i' )
```
retournera la chaîne `"surpr"` (*string*)
```
extrait( "Quelle surprise", 7, 'a' )
```
retournera la chaîne `"surprise"` (*string*)

Question 2

Écrivez une fonction `normalise` qui prend en paramètres une chaîne de caractères `chaine` (la fonction manipulera la chaîne de caractère d'origine, protégée de toute modification), un entier `longueur` et un caractère `c` (qui par défaut sera initialisé au caractère espace). Cette fonction renvoie une chaîne obtenue à partir de `chaine` de la façon suivante :

- si la longueur de `chaine` est supérieure à `longueur`, elle tronque `chaine` à `longueur` ;
- si la longueur de `chaine` est inférieure à `longueur`, elle complète `chaine` par des caractères `c` jusqu'à obtenir une chaîne dont la taille est `longueur`.

```
normalise( "Quelle surprise" , 6 , 'e' )
```
retournera la chaîne `"Quelle"`
```
normalise( "Quelle surprise" , 20 , 'X' )
```
retournera la chaîne `"Quelle surpriseXXXXX"`

Question 3

Écrivez une fonction `compacte` qui prend en paramètres une chaîne de caractères `s` et un caractère `c` (qui **par défaut** sera initialisé au caractère espace) et qui retourne une chaîne de caractères obtenue à partir de `s` de la manière suivante : toute suite de caractères identiques à `c` est remplacée par un seul `c`.

```
compacte( "Quelllle surprise" , 'l' )
        retournera la chaîne "Quele surprise"
compacte( "   Quelllle    surprise" )
        retournera la chaîne " Quelllle surprise"
```

Exercice 3 (subsidiaire) : Les palindromes

Palindrome *:*
se dit d'un mot ou d'une phrase qu'on peut lire dans les deux sens,
en commençant par la droite ou par la gauche :
« Esope reste ici et se repose ! » ou « ressasser » ou encore « engage le jeu que je le gagne » .

Proposez une ou plusieurs fonctions permettant de décider si une chaîne de caractères est un palindrome. Le cas des caractères non alphabétiques n'est pas à considérer. Réfléchissez dans un premier temps à un découpage en fonctions, déclarez leur prototype puis proposez une fonction `main` les utilisant. Enfin donnez la définition de l'ensemble des fonctions déclarées auparavant, compilez et testez votre programme.

Manipulation de l'objet *string*

Création

```
string chaine1;                          → chaine1 : chaîne vide
string chaine2 ( "bonjour" );            → chaine2 : bonjour
string chaine3 ( "bonjour" , 4 );        → chaine3 : bonj
string chaine4 ( 5 , 'X' );              → chaine4 : XXXXX
string chaine5 = "onjou";                → chaine5 : onjou
```

Longueur

```
int taille = chaine2.length();           → taille  : 7
```

Saisie

Dans les exemples qui suivent `ch1`, `ch2` et `ch3` sont de type *string*, et la chaîne de caractères que saisit l'utilisateur est : `Ah le beau samedi`

```
cin >> ch1;                              → ch1  : Ah
getline( cin , ch2 );                    → ch2  : Ah le beau samedi
getline( cin , ch3 , 'u');               → ch3  : Ah le bea
```

Recherche de la position d'un caractère

```
int ind1 = chaine5.find('o');            → ind1 : 0
int ind2 = chaine5.find('o',1);          → ind2 : 3
int ind3 = chaine5.find('r',1);          → ind3 : -1
int ind4 = chaine5.rfind('o');           → ind4 : 3
int ind5 = chaine5.rfind('o',2);         → ind5 : 0
int ind6 = chaine5.rfind('r');           → ind6 : -1
int ind7 = chaine5.find_first_not_of('o');   → ind7 : 1
int ind8 = chaine5.find_first_not_of('o',3); → ind8 : 4
int ind9 = chaine5.find_first_not_of('r');   → ind9 : 0
```

Divers

```
char caract = chaine2[3];                → caract    : j
string ssChaine2 = chaine2.substr( 2, 4 );   → ssChaine2 : njou
string ssChaine3 = chaine3.substr( 2, 4 );   → ssChaine3 : nj
string ssChaine4 = chaine2.substr( 2 );      → ssChaine3 : njour
chaine1 = chaine2 + chaine4;             → chaine1   : bonjourXXXXX
chaine1 = chaine2 + 'R';                 → chaine1   : bonjourR
chaine4 += 'R';                          → chaine4   : XXXXXR
chaine2.erase( 2 , 3 );                  → chaine2   : bour
```