Arduino
Savoir utiliser la classe String

La classe String permet de manipuler simplement les chaînes de caractères pour afficher des informations (sur le port série ou dans un fichier par exemple).
On peut également convertir des nombres en chaînes et inversement.
Voir les nombreuses possibilités ici :
https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/

```
#include "Arduino.h"
String msg = "";
String number="345";
String systemStatus = "heater:ON;fan:OFF" ;
int n = 12, p;
float x = 15.22458 , z;

void setup(){
    Serial.begin(115200);
    Serial.print("\n\n\n\n");
    msg = "n= " + …….;
    Serial.print(msg);
    msg = …….;
    Serial.print(msg);
    p = ……. + 1 ;  // augmenter number de 1
    Serial.println(p);
    z = ……. + 0.5 ; // augmenter number de 0.5
    msg = …….;
    Serial.println(msg);
    Serial.print("le chauffage est sur ");
    msg = …….;  // utiliser la variable info
    Serial.print(msg);
    Serial.println();       // utiliser la variable info
}

void loop(){
}
```

Complétez le programme ci-dessus pour produire exactement l'affichage suivant :

n=12 x=15.2246
n vaut 1100 en binaire et c en hexa
346
z=345.50
le chauffage est sur ON le ventilateur est sur OFF

Exploitation d'une trame GPS NMEA1083

Un GPS délivre des trames au format NMEA1083 sur un port série.

Par exemple la trame :

```
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,,0000,*22
```

est décodé ainsi :

$GPGGA : Type de trame
064036.289 : Trame envoyée à 06 h 40 min 36 s 289 (heure UTC)
4836.5375,N : Latitude 48,608958° Nord = 48° 36' 32.25" Nord
00740.9373,E : Longitude 7,682288° Est = 7° 40' 56.238" Est
1 : Type de positionnement (le 1 est un positionnement GPS)
04 : Nombre de satellites utilisés pour calculer les coordonnées
3.2 : Précision horizontale ou HDOP (*Horizontal dilution of precision*)
200.2,M : Altitude 200.2, en mètres
,,,,,0000 : D'autres informations peuvent être inscrites dans ces champs
22 : Somme de contrôle de parité, un simple XOR sur les caractères entre $ et * ($ et * exclus).

Pour exploiter facilement cette trame on définit la structure suivante :

```
struct gpsData {

 String type ; // sans le $
 String time ; // format HH:MM:SS
 float timeMs ; // millisecondes supplémentaires de
time
 double lat ; // >0 N <0 S en degrés décimaux
 double lon ; // >0 E <0 W en degrés décimaux
 bool posGps ; // vrai si positionnement GPS faux
sinon
 int satNum ; // nombre de satellites utilisés
 float hdop ; // précision horizontale en mètres
 float altitude ; // en mètre
 uint8_t cheksum ; //  checksum en binaire
 bool cheksumOk ; // vrai si cheksum correcte

} ;
```

La trame reçue est stockée dans une variable de type **String**.
On veut remplir la structure avec les données de la trame.
Pour cela on écrit une fonction :

```
void fillData( String trame, struct gpsData& data ) ;
```

Le & indique un passage par "référence" : cela permet de voir data comme un paramètre passé par valeur (et donc d'utiliser le . pour accéder aux champs) tout en obtenant la modification dans la fonction appelante comme si on avait passé son adresse avec un pointeur.

On définit une autre fonction pour imprimer :

```
void printData( struct gpsData data);
```

Avec la trame d'exemple on obtiendrait :

```
Trame reçue :
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,2
00.2,M,,,,,,0000,*22
type : GPGGA
time : 06:40:36
timeMs :0.289
lat : 48.608958
lon : 7.682288
gps mode : 1
sat number : 4
horiz. dilution : 3.20
checksum : 0x22
checksum ok : 1
```

Complétez le programme de la page suivante pour remplir la structure et l'afficher.

Il faut procéder par étape en progressant dans la trame. Il peut être judicieux de compléter le remplissage de la trame en même temps que l'affichage.
Exploitez les possibilités de la classe String. Notamment les méthodes indexOf(), substring() , toDouble() et toInt().

Rappel : **conversion degré, minutes,secondes en degré décimaux**
Le degré est équivalent à l'heure, la minute d'arc à la minute, la seconde d'arc à la seconde donc 1° = 60' = 3600"
48° 36' 32.25" :
36 minutes d'arc = 36/60 degrés
32.25 secondes d'arc = 32.25/3600 degrés
(la partie fractionnaire des secondes est toujours décimale)
Donc 48° 36' 32.25" = 48 + 36/60 + 32.25/3600 = 48,608958 degrés

Mais le GPS donne une fraction décimale de minute et pas des minutes,secondes donc 4836.5375 correspond à 48° et 36.5375 minutes d'arc. Donc la conversion est simplement :
48.365735= 48 + 36.5735/60 = 48,608958 degrés décimalisés.

```
struct gpsData {
 String type ; // sans le $
 String time ; // format HH:MM:SS
 float timeMs ; // millisecondes supplémentaires de time
 double lat ; // >0 N <0 S en degrés décimaux
 double lon ; // >0 E <0 W en degrés décimaux
 bool posGps ; // vrai si positionnement GPS faux sinon
 int satNum ; // nombre de satellites utilisés?
 float hdop ; // précision horizontale en mètres
 float altitude ; // en mètre
 uint8_t cheksum ; // en binaire
 bool cheksumOk ; // vrai si cheksum correcte

} ;
//String
trame2="$GPGLL,5300.97914,N,00259.98174,E,125926,A*28";
String trame1 =
"$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,
,,,,0000,*22";
struct gpsData data1 ;
void fillData( String trame, struct gpsData& data ) ;
void printData( struct gpsData data);

void setup() {
 Serial.begin(115200);
 Serial.print("\n\n\n\n");
 fillData(trame1, data1);
 Serial.println("Trame reçue " + trame1 ) ;
 printData(data1);
}

void loop() {

}

void fillData( String trame, struct gpsData& data) {

}
void printData( struct gpsData data) {
 Serial.println("type : " + data.type) ;
 Serial.print("time : "); Serial.println( );
 Serial.print("timeMs :" );
 Serial.print("lat : ");
 Serial.print("lon : ");
 Serial.print("gps mode : ");
 Serial.print("sat number :");
 Serial.print("horiz. dilution :");
 Serial.print("checksum : ");
}

bool verifyChksum(String trame ) {

 }
```