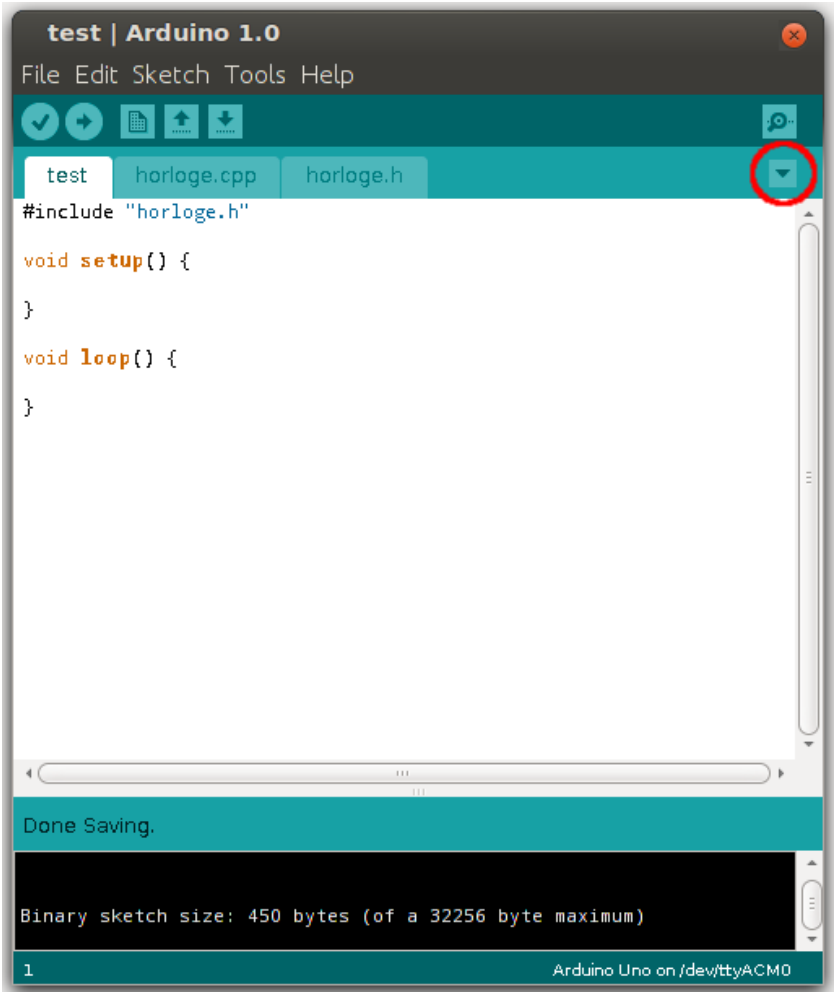
Modularité

1. Organiser votre code en fichiers

Lorsque vous commencez à faire de gros projets, il devient utile voire indispensable de (très) bien organiser son code. Cela commence par séparer son code en différents fichiers afin d'avoir des entités logiques séparées les unes des autres.

Une opération simple à faire et qui permet de gagner beaucoup en organisation de son code est de séparer ce dernier en différents fichiers. Généralement, on fait un fichier par unités "logiques". Par exemple, imaginons que nous utilisions un composant un peu compliqué qui sert d'horloge. Ce composant peut renvoyer une date en entier, juste le jour, mois, année ou encore juste l'heure, la minute ou la seconde courante. Pour bien faire, il nous faudrait une fonction par unité de temps. On aurait ainsi au moins 6 fonctions pour récupérer heure/minutes/secondes/jour/mois/année et 6 fonctions pour les régler dans le composant.

Pour créer un nouveau fichier dans l'IDE Arduino, il suffit de cliquer sur la petite flèche en haut de l'espace d'édition du code puis ensuite de cliquer sur "Nouvel Onglet" ou "New Tab" comme mis en évidence sur la capture d'écran ci-dessous :



2. Le fichier .h

lorsque l'on veut séparer son code en plusieurs fichiers, il y a certaines choses à respecter. Ainsi, à chaque fois que l'on veut créer un nouveau fichier de code on ne vas pas en créer un mais deux ! Le premier fichier aura l'extension .h signifiant **header**, c'est ce que nous allons voir maintenant.

Ce fichier va regrouper les **prototypes** des fonctions ainsi que les définitions de structures ou de classes mais nous verrons cela après.

Le prototype d'une fonction représente un peu un contrat. Il va définir le nom de la fonction, ce qui rentre à l'intérieur (les paramètres) et ce qui en sort (la variable de retour). Ainsi, votre programme principal aura une idée de comment fonctionne *extérieurement* votre fonction. Un peu comme s'il s'adressait à une boîte noire.

Si l'on devait écrire l'exemple ci-dessus on pourrait avoir le contenu de fichier suivant :

horloge.h

```
char getHeure();
char getMinute();
char getSeconde();
char getJour();
char getMois();
char getAnnee();

void setHeure(char val);
void setMinute(char val);
void setSeconde(char val);
void setJour(char val);
void setMois(char val);
void setAnnee(char val);

void afficherDate();
void afficherHeure();
void afficherDateHeure();
```

Comme vous pouvez le voir, avec ces définitions on peut savoir ce qu'est supposé faire la fonction grâce à son nom et le type de variable qu'elle manipule en entrée et en sortie.
Bien, maintenant passons à la suite pour voir où et comment implémenter ces fonctions.

2. Le second fichier .cpp

Le second fichier que nous allons créer sera avec une extension .cpp (pour C plus plus ou C++). Il regroupera le code à proprement parler, l'implémentation de vos fonctions. C'est ici que vous allez écrire le contenu de vos fonctions, ce qui est censé se passer à l'intérieur de ces dernières.

Pour faire cela, la première étape sera d'inclure le fichier de prototypes via la commande de préprocesseur `#include` :

```
#include "horloge.h" // horloge.h pour notre exemple
```

Cette ligne doit être la **première** de votre fichier .cpp et elle ne prend pas de `;` à la fin.

Une fois cela fait, il va falloir taper le code de vos fonctions.

horloge.cpp

```
/* fichier horloge.cpp */
#include "horloge.h"

char getHeure()
{
   Serial.println("getHeure");
   return 0;
}

char getMinute()
{
   Serial.println("getHeure");
   return 0;
}

char getSeconde()
{
   Serial.println("getHeure");
   return 0;
}

char getJour()
{
   Serial.println("getHeure");
   return 0;
}

char getMois()
{
   Serial.println("getHeure");
   return 0;
}

char getAnnee()
{
   Serial.println("getHeure");
   return 0;
}

void setHeure(char val)
{
   Serial.print("setHeure : ");
   Serial.println(val, DEC);
}

void setMinute(char val)
{
   Serial.print("setMinute : ");
   Serial.println(val, DEC);
}

void setSeconde(char val)
{
   Serial.print("setSeconde : ");
```

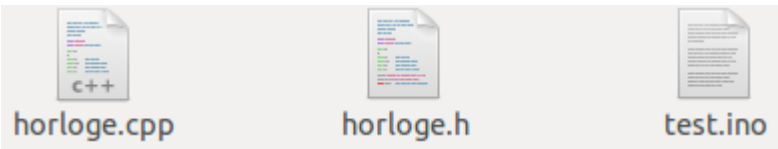
```
    Serial.println(val, DEC);
}

void setJour(char val)
{
    Serial.print("setJour : ");
    Serial.println(val, DEC);
}

void setMois(char val)
{
    Serial.print("setMois : ");
    Serial.println(val, DEC);
}

void setAnnee(char val)
{
    Serial.print("setAnnee : ");
    Serial.println(val, DEC);
}

void afficherDate()
{
    Serial.println("afficherDate");
}

void afficherHeure()
{
    Serial.println("afficherHeure");
}

void afficherDateHeure()
{
    Serial.println("afficherDateHeure");
}
```

4. Lier nos fichiers au programme principal

Vos définitions sont écrites et vos fonctions sont implémentées ? Il ne reste plus qu'à les ajouter à votre programme principal !

Tout d'abord, il va falloir s'assurer que vos fichiers .h et .cpp sont dans le même dossier que votre .ino où se trouve votre fichier de programme Arduino.

Comme ceci :



Il faut ensuite l'inclure dans le programme. Pour cela, il suffit d'ajouter la ligne `#include "horloge.h"` en haut de votre fichier.

Il ne vous reste plus qu'à faire des appels tout simples à vos fonctions perso dans le programme (setup ou loop ou où vous voulez !).

Maintenant, quand vous allez compiler, le compilateur va aller chercher le fichier pointé par le include, le compiler puis le lier dans votre programme principal.

Il peut arriver que le lien avec les symboles/librairies Arduino ne se fasse pas correctement. Dans ce cas là, rajoutez l'include suivant au début de votre .h ou .cpp : `#include "Arduino.h"`