

Introduction

Dans le cadre du module « Algorithmique et Programmation » de 1^{ère} année, il est utile d'être en capacité de mettre en œuvre les implémentations d'algorithme vu en TD (ou lors de la séance de TP) afin de tester la compréhension du fonctionnement du langage C en particulier et des algorithmes en général.

Réalisation pratique :

- Ecriture et exécution d'un premier programme pas à pas à l'aide de l'interface du logiciel Code::Blocks ;
- Mise en œuvre d'implémentation dans divers contextes ;
- Appropriation et correction d'un code source à l'aide des outils de base du compilateur et debugger ;

Objectifs du TP :

- Prendre en main Code::Blocks et les base du langage C ;
- Exploiter une implémentation en sous-programme ;
- Adopter les bonnes pratiques de programmation (nom de variables, commentaires, etc) ;

Attendu du compte-rendu :

- Fournir l'ensemble du code source effectué lors du TP ;

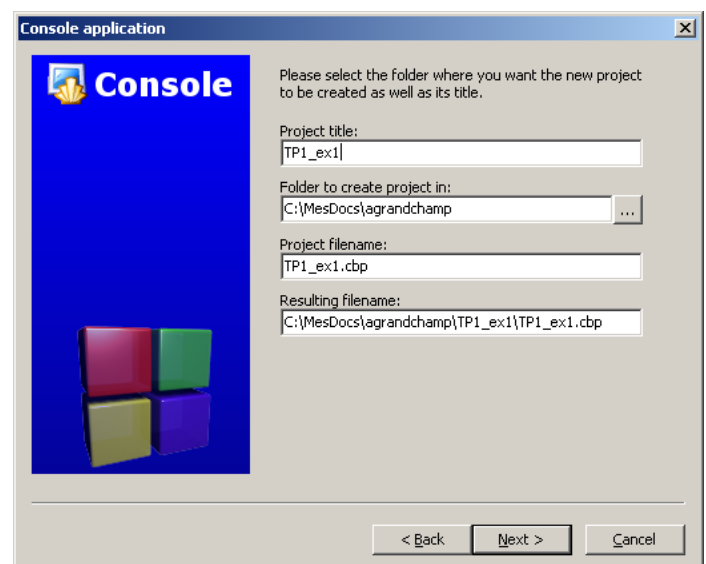
Mise en route pas à pas d'un premier programme sous Code::Blocks

Suivez pas à pas les étapes marquées d'un tiret « - » ci-après.

Création du projet

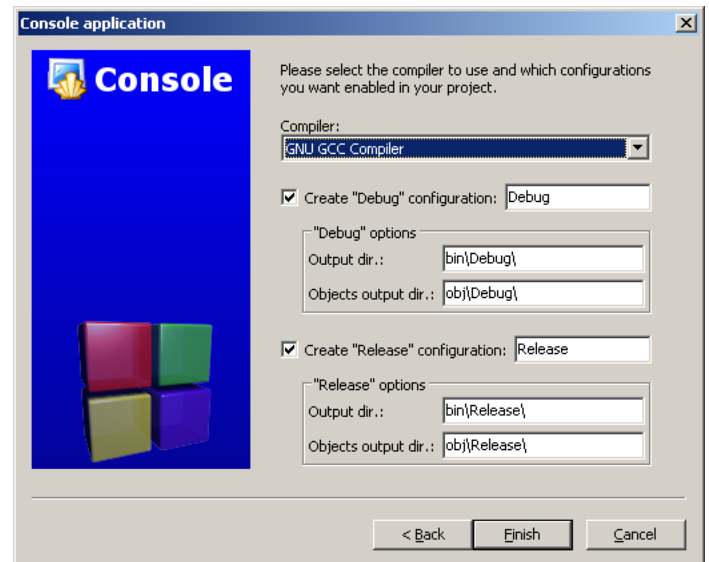
Pour chaque partie du TP :

- File/New/Project...
- Choisir Console Application puis Go
- Next
- Choisir le langage C (et non C++)
- Next



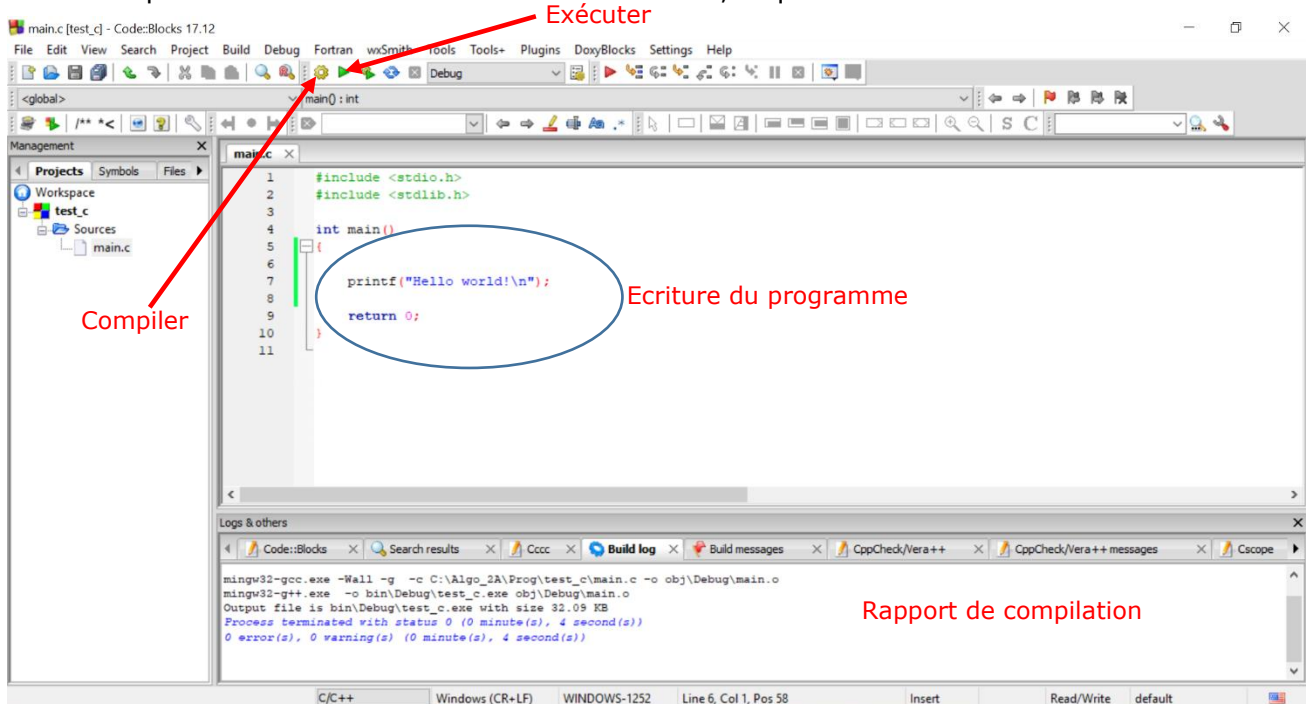
- Bien choisir le nom du projet (sans espace, sans accents) et bien choisir le dossier dans lequel ranger le projet
- Next

- Cliquer sur « finish », le projet est créé !



Ecriture du code source

- Cliquer sur le + à côté du dossier « Sources », cliquez sur « main.c » :



1 Retrait d'argent (environ 30 – 45 minutes)

- 1) Ecrire un programme qui demande la somme d'argent souhaitée et détermine si le retrait est autorisé ou non. Le retrait est refusé dans le cas où la somme demandée dépasse 100. Le message sera adapté selon la situation : « Retrait accepté » ou « Retrait refuse ».

```
Bonjour, entrez la somme demandee : 130
la somme demandee est : 130
retrait refuse
```

Figure 1 : sortie console

- 2) Modifier le programme pour permettre de demander plusieurs fois de retirer une somme, le programme s'arrête quand l'utilisateur indique 0.

Sortie console du programme :

```
Bonjour,  
entrez la somme demandee : 130  
la somme demandee est : 130  
retrait refuse  
entrez la somme demandee : 90  
la somme demandee est : 90  
retrait autorise  
entrez la somme demandee : 0  
au revoir
```

Figure 2 : sortie console

- 3) Nous allons créer un sous-programme :

- Copiez le prototype du sous-programme ci-dessous entre les instructions pré-compilateur et la ligne « int main() » :

```
void afficherAutorisationRetrait(int montantRetrait);
```

- Créez l'en-tête de la définition de ce sous-programme après la dernière accolade de votre programme ;
- Coupez-collez la partie du code concernant l'affichage de l'autorisation de la question 1 entre les accolades du sous-programme ;
- Effectuez un appel au sous-programme dans votre programme principal ;

2 Introduction aux sous-programmes (environ 1h15 – 1h30)

I) Nous proposons d'écrire un programme C (sans utiliser de sous-programmes) qui permet de calculer le factoriel d'un nombre.

1. Identifier les entrées ainsi que les sorties de ce programme.
2. Ecrire puis tester un programme C qui calcule le factoriel d'un nombre.

Nous souhaitons écrire un programme C qui permet de calculer un coefficient binomial $\binom{n}{k}$.

3. Identifier les entrées et les sorties de ce programme.
4. Combien de fois serait-il nécessaire d'écrire le code qui permet de calculer un factoriel ? Est-ce raisonnable ? Quelle solution de programmation vous permettez d'écrire une seule fois l'algorithme de calcul du factoriel ?

II) Nous allons donc maintenant utiliser un sous-programme pour alléger le code.

Nous commencerons par écrire une fonction « fact » qui permet de calculer le factoriel d'un nombre.

5. Identifier le(s) entrée(s) ainsi que le(s) sortie(s) de ce sous-programme et proposer un prototype pour cette fonction au-dessus du programme principal.
6. Implémenter la définition complète de cette fonction en dessous du programme principal.
7. Ecrire dans le programme principal un appel à cette fonction avec la valeur 5 et affiche le résultat pour vérifier que le sous-programme est correct.
8. Compléter le programme principal qui demande n et k à l'utilisateur et affiche le résultat $\binom{n}{k}$ à l'écran (vous utiliserez autant de fois que nécessaire votre fonction « fact »).

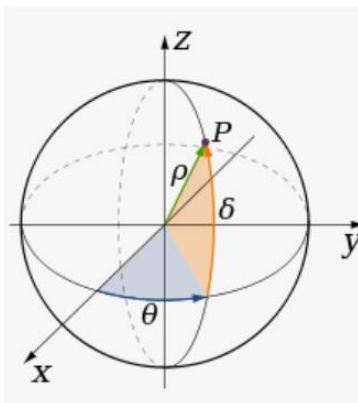
III) Enfin, nous voulons garantir que nous effectuons un calcul $\binom{n}{k}$ avec $k < n$ quels que soient les saisies utilisateurs.

9. Ecrire un sous-programme qui demande à l'utilisateur deux entiers et qui détermine le plus petit ainsi que le plus grand de ces deux entiers (qui sont les deux sorties respectives). Le prototype est donné ci-dessous :

```
void saisirMinMax(int *min, int *max);
```

10. Intégrer l'utilisation de ce sous-programme dans le programme principal.

3 Ecriture de sous-programme pour répondre au CdC proposé (~1h30)



Les coordonnées GPS sont des coordonnées sphériques exprimées en degré (dans notre cas degré décimal). Sur la Figure 3, le point P a pour coordonnées GPS (δ , θ) où δ est la **latitude** et θ représente la **longitude**. Sur le schéma ρ désigne le rayon de la sphère. Dans la suite de l'exercice nous considérerons que le **rayon de la terre** vaut 6371 km.

Pour calculer la distance entre deux points A et B positionnés sur la sphère, il est possible d'utiliser la formule ci-dessous :

$$d = \rho \arccos [(\cos(\theta_A - \theta_B) \cdot \cos \delta_A \cdot \cos \delta_B + \sin \delta_A \cdot \sin \delta_B)]$$

Figure 3 : Coordonnées sphériques

Par exemple, il y a 393 km entre Paris Lat 48.85° Long 2.35° et Lyon Lat 45.75° Long 4.85°. Vous testerez systématiquement le bon fonctionnement des sous-programmes.

3.1 Implémentation de fonctions utiles

- Sous-programmes de calcul (**pas de printf/scanf !**)

Implémenter les deux fonctions effectuant les calculs précédemment décrits dont les prototypes vous sont proposés ci-dessous :

```
float convertDegToRad(float valeurDegre);
float calculDistance(float latA, float longA, float latB, float longB);
```

Rmq : les longitudes et latitudes sont exprimées en degrés en paramètre. Les fonctions de la bibliothèque <math.h> trigonométriques attendent des valeurs en radians.

- Sous-programmes d'interaction (**printf/scanf**)

Implémenter la fonction suivante qui interagit avec l'utilisateur pour lui demander une valeur **tant qu'elle n'est pas comprise** dans l'intervalle [borne Inf ; borneSup] puis renvoie cette valeur.

```
float saisirValBornee(int borneInf, int borneSup);
```

Implémenter la procédure suivante permettant de saisir les valeurs de longitude et de latitude d'un point. L'objectif de cette fonction est d'interagir avec l'utilisateur pour lui demander les valeurs à prendre en compte. Prendre garde à ce que l'utilisateur ne puisse saisir que des valeurs comprises entre -90 et +90 pour la latitude et entre -180 et +180 pour la longitude.

```
void recupCoordonnees(float *latitude, float *longitude);
```

Dans ce sous-programme, vous pouvez utiliser votre sous-programme « saisirValBornee ».

3.2 Construction du programme

Implémenter le sous-programme « saisirVoyage » de manière à ce que son utilisation proposée dans le « main » ci-dessous soit vérifiée. Cette fonction saisirVoyage s'appuyera sur les fonctions et la procédure développées précédemment.

```
int main()
{
    //declaration des variables
    float distance;

    distance = saisirVoyage();
    printf("La distance du voyage est : %f\n",distance);

    return 0;
}
```

3.3 Modification du programme

Ajouter un sous-programme qui permet de déterminer le moyen de transport approprié en fonction de la distance calculée. Ce programme dessinera le moyen de transport choisi selon les propositions ci-dessous.

A pied :

```
Oooo.
(  /
 ) /
(  /
 -
```

A vélo :

```
      O
      |
  <---|
  (*)/  (*)
```

En voiture :

```
  /  \
 /____\
|       |
|  O  O  |
```

4 Correction de programme fourni (environ 45 min – 1h)

- Fermer tous les précédents projets
- Copier-coller le dossier fourni AutoPixelArt
- Ouvrez le projet à l'aide de Code::Blocks AutoPixelArt.cbp

Vous allez corriger les différentes erreurs du programme fourni à l'aide des outils de code::blocks

4.1 Erreurs syntaxiques

Il y a 3 erreurs syntaxiques qui vont être détectées par le compilateur Code::Blocks à corriger.

- Lancez le compilateur (Ctrl-F9) pour qu'il vous donne le message d'erreur correspondant et la ligne de l'erreur détectée ;

- Effectuez le correctif attendu dans le fichier Main_APA.c en vous appuyant sur le message et le tableau d'aide ci-dessous ;
- Répétez l'opération 3 fois jusqu'à ne plus avoir d'erreur renvoyée par le compilateur ;

Message du compilateur	Traduction	Correction à apporter
error: expected ';' before [...]	Le compilateur s'attendait à un point-virgule	Il manque avant la ligne d'erreur le point-virgule
error: conflicting types for [...]	Types conflictuels pour la fonction désigné	Les arguments en déclaration du prototype et définition ne sont pas les mêmes
expected expression before [...]	Une instruction est incomplète	Il manque juste avant une instruction correctement écrite

4.2 Erreurs sémantiques

Il y a 2 erreurs sémantiques qu'il faut détecter en interprétant la différence entre ce que fait effectivement le programme et le but recherché.

Nous allons d'abord tester la bonne exécution de la fonction.

Pour cela :

- Rajoutez `#define TEST 1` à la ligne 1 du code ;
- Rajoutez `#if TEST` puis `#else` sur deux lignes avant « `int main()` » ;
- Déplacez-le `#endif` après le bloc « `main` » ;

Nous pouvons maintenant implémenter un nouveau code « `main` » qui va tester le fonctionnement de la fonction proposée.

- Copiez-collez entre les lignes `#if` et `#else` le code ci-dessous :

```
int main()
{
    printCenteredSymbol('T',5,CENTRE_CONSOLE);
    printCenteredSymbol('T',3,CENTRE_CONSOLE);
    printCenteredSymbol('T',1,CENTRE_CONSOLE);
    return 0;
}
```

- Compilez et exécutez le code (touche F9) et dites si le sous-programme réalise la fonction attendue d'écrire 5, puis 3 puis 1 « T » centrés sur la console ;
- Pour corriger le problème, concentrez-vous sur la ligne du code du sous-programme qui est donnée ci-dessous et changez la pour corriger le problème :

```
for(i=0; i<center-nbSymbol; i++) printf(" ");
```

Le problème dans le sous-programme est ainsi normalement corrigé.


- Changez votre première ligne de `#define TEST 1` à `#define TEST 0` afin de repasser sur le programme « `main` » normal ;

Nous allons utiliser l'outil « Pas à Pas » du debugger pour trouver la deuxième erreur sémantique.

- Cliquez juste à côté du numéro de ligne du code et celui d'en dessous :

```
for(i=LARGEUR_HAUT; i<11; i=i+2); printCenteredSymbol('^',i,CENTRE_CONSOLE);
```

Cela rajoute ce qu'on appelle un « Breakpoint » ou point d'arrêt. Il va être ainsi possible d'observer le programme en cours d'exécution plutôt que voir uniquement l'exécution complète.

- Exécutez le programme en mode pas à pas (F8) ;
- Ouvrez les observateurs (icône  -> « watches ») pour observer les valeurs des variables en mémoire au moment de l'arrêt de l'exécution ;

A chaque appui sur Alt+F7, une exécution d'une instruction est faite (càd ici à chaque itération de la boucle).

- Observez l'incrémentation de la variable `i` et observez l'affichage sur la console ;

Le code effectue-t-il ce qui est attendu ? Corrigez alors l'erreur se trouvant dans cette ligne.

4.3 Bonnes pratiques

Ecrire un code correct passe aussi par la forme que vous lui donnez. Le travail s'effectue généralement en équipe et nécessite de communiquer aussi par la manière dont vous écrivez votre code.

Essayez les fonctions suivantes pour améliorer la mise en forme du code :

- Menu « Plugins » ⇒ « Source Code Formatter » pour mettre en forme de manière claire et automatique votre code.
- Cliquez sur la ligne de déclaration de la fonction puis menu « DoxyBlocks » ⇒ « Block Comment » pour générer des commentaires automatiques de la fonction : complétez en décrivant ce que réalise la fonction et le rôle de chaque argument ;
- Enlevez toutes les valeurs du code qui sont censés être en constante de pré-compilateur comme l'exemple de `LARGEUR_HAUT` et `LONGUEUR_HAUT` du code initial ;

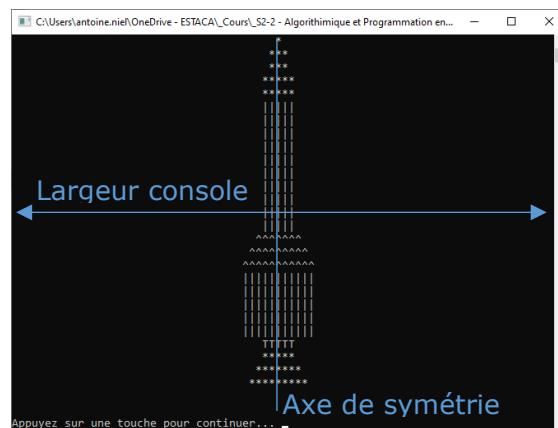


Figure 4 : résultat de console lorsque le code fonctionne correctement