Polytech Lille
IS2A – 3ème Année

TP PROGRAMMATION C
Compilation - Passage de paramètres

Pour démarrer :

- Ouvrez un terminal
- Créez un répertoire progC (ou tout autre nom à votre convenance !) à la racine de votre répertoire privé
- Positionnez vous dans ce répertoire
- Lancez votre éditeur préféré en arrière plan

Note : le lancement de commande en arrière plan permet au shell de vous redonner la main sans attendre que la commande lancée soit terminée. Pour lancer une commande en arrière plan, il suffit de terminer la commande par le caractère &. Exemple : la commande permettant de lancer l 'éditeur kate en arrière plan est kate &.

Un premier programme C :

Mais avant, rappelez-vous les « bonnes règles » de programmation C !!!

règle 1 : Jamais de constantes en dur dans les programmes. Les constantes sont définies par #define
règle 2 : Le code est correctement indenté pour simplifier la lecture
règle 3 : On donne des noms significatifs aux variables et fonctions pour la lisibilité du programme
règle 4 : On commente le code chaque fois que nécessaire pour la lisibilité
règle 5 : On n'utilise pas de variable globale … pour l'instant
règle 6 : On respecte les conventions de nommage (constantes en MAJUSCULES, identificateurs de fonctions et variables en minuscules)
règle 7 : Les variables de bloc sont déclarées en début de bloc
règle 8 : On ne fait pas d'économies de « bouts de chandelles » au détriment de la lisibilité du programme
règle 9 : On n'oublie jamais les règles 1 à 8 !

- Ecrire une fonction *affichage*, affichant la valeur de 3 réels x, y et z donnés
- Ecrire une fonction *lecture* réalisant la lecture au clavier de 3 réels
- Ecrire la fonction *main* permettant de tester les 2 fonctions précédentes

Prêt à compiler ?

Pour produire un exécutable à partir d'un source C, on utilise l'utilitaire **gcc**.

gcc enchaîne automatiquement toutes les phases de production de l'exécutable

- appel au préprocesseur C qui traite les lignes commençant par # et produit un source C étendu (il remplace les lignes #include par le contenu des fichiers .h, il substitue toutes les occurrences des constantes définies par #define par l'expression associée)
- appel au compilateur qui effectue l'analyse lexicale et l'analyse syntaxique du code C étendu et s'il n'y a pas d'erreur, produit le code assembleur
- appel à l'assembleur qui traduit le code assembleur en code binaire : à ce stade est produit un fichier d'extension .o (un fichier objet) qui contient du code exécutable (binaire) « incomplet » ou un code dit « à trou » : il manque en particulier le code exécutable de toutes les fonctions de bibliothèque qui ont été utilisées dans le source C (printf, scanf, … par exemple), on appelle les « trous » des références manquantes.

Rappel : l'inclusion de fichiers entête dans le source C (exemple : #include <time.h>) permet d'inclure la signature de fonctions (uniquement l'entête), ce qui permet au compilateur de vérifier que l'on fait un bon usage de ces fonctions ; le code de la fonction n'est pas inclus par cette directive.

- appel à l'éditeur de lien qui « rassemble tous les morceaux » dans le but de « combler les trous » et produire le fichier exécutable : par défaut, il recherche toutes les références manquantes dans la bibliothèque C standard (fichier de nom libc.a). Une bibliothèque est un archivage de fichier objet (fichier d'extension .o). Ces fichiers contiennent le code exécutable de fonctions, des définitions de type, de constantes etc … Lorsque l'éditeur de lien trouve une référence manquante dans un fichier objet d'une bibliothèque, il inclut ce fichier objet dans le fichier exécutable. A l'issue de la phase d'édition de lien, s'il reste des références manquantes, une erreur d'édition de lien est générée précisant quelle référence manque, sinon le fichier exécutable est produit, on peut tester le programme !!

Mais quelle est la commande de compilation ?

Si votre source s'appelle programme.c et que vous souhaitez produire un fichier exécutable de nom programme, la commande à lancer à partir du shell est la suivante :

gcc -Wall -o programme programme.c

-Wall est une option donnée au compilateur permettant d'obtenir tous les « warnings » de compilation. Règle de base : un warning doit toujours donner lieu à une correction du code.

-o est une option donnée à l'éditeur de lien qui permet de préciser le nom de l'exécutable à produire

Note : On verra plus tard les commandes de compilation séparée ou comment indiquer à l'éditeur de lien qu'il doit rechercher les références manquantes dans d'autres bibliothèques que la bibliothèque standard.

C'est parti !

Compilez votre source C, corrigez les erreurs et **TOUS** les warnings, testez l'exécution.

On complète le programme

- Ecrire une fonction *minmax2* qui range le minimum de 2 réels a et b dans a et le maximum dans b
- Etant donnés 3 réels a, b, c, écrire une fonction *tri* qui utilise la fonction minmax2 pour ranger le minimum des 3 réels dans a, le maximum dans c et le 3ème réel dans b.
- Compléter la fonction *main* pour tester la fonction tri (on utilisera la fonction affichage précédemment définie pour afficher les valeurs « triées »

Quelques exercices pour continuer !

1 Manipulation de caractères

- Ecrire une fonction *chiffre* déterminant si un caractère ch donné est un chiffre
- Ecrire une fonction *minuscule* déterminant si un caractère ch donné est une lettre minuscule

- Ecrire une fonction *majuscule* déterminant si un caractère ch donné est une lettre majuscule
- Ecrire une fonction *analysePhrase* demandant à l'utilisateur une suite de caractères terminée par le caractère '.' et déterminant le nombre de caractères, de chiffres, de lettres minuscules, et de lettres majuscules saisis par l'utilisateur
- Tester ces fonctions dans un main

2 Manipulation d'entiers

- Ecrire une fonction *lireDate* permettant de lire une date sous la forme jj mm aaaa.
- Ecrire une fonction *lire2Dates* qui utilise la fonction lireDate pour lire 2 dates sous la forme jj mm aaaa
- Ecrire une fonction *compareDates* qui étant données 2 dates d1 et d2 retourne 0 si d1=d2, -1 si d1<d2 et 1 sinon
- Ecrire la fonction main qui lit 2 dates et affiche le résultat de leur comparaison

3 Suite d'entiers

- Ecrire une fonction *somme* qui détermine la somme de 2 réels données
- Ecrire une fonction *minMax* qui pour 3 réels a, b et c données, détermine le minimum et le maximum
- Ecrire une fonction *traitementSuite* demandant à l'utilisateur une suite de réels terminée par 0 et calculant, à l'aide des sous-programmes précédents, la moyenne, le minimum et le maximum de cette suite
- Ecrire la fonction main qui appelle la fonction *traitementSuite* puis affiche la moyenne, le minimum et le maximum de la suite saisie par l'utilisateur

Il vous reste du temps ?

Soundex code (Enfin un peu d'algorithmique !!)

Le soundex code est utilisé par l'administration américaine pour regrouper sous un même code les noms ayant des consonances voisines.

Le code que nous considérons ici en est une version simplifiée, il associe à un nom l'abréviation obtenue en appliquant les 4 règles suivantes :

1) Conserver la première lettre du nom
2) Supprimer toutes les voyelles
3) Remplacer les lettres doubles par une seule
4) l'abréviation a 4 lettres maximum

Exemples :

- DUPONT => DPNT
- LLOYD => LD
- RASTATOPOULOS => RSTT

Ecrire un programme qui lit une suite de caractères au clavier (terminée par un caractère autre qu'une lettre) et affiche à l'écran le soundex code du mot saisi.