ENSEIRB 2006-2007 - Première année
PG101 - Programmation en C

TD 1 : Premiers programmes & compilation

Le but de cette feuille de TD est d'une part d'appréhender la syntaxe du langage C au travers de premiers programmes, en particulier, les notions élémentaires sur les variables et leurs types. D'autre part, il s'agit de compiler et d'exécuter ces programmes, et d'aborder les premiers problèmes de compilation, avertissements et erreurs, et leur correction.

Créez un répertoire nommé `TDc/` et un sous-répertoire `TDc/td1/` dans votre répertoire personnel (avec la commande `mkdir` dans un terminal). Positionnez-vous dans ce sous-répertoire (avec la commande `cd`), vous y resterez toute cette séance.

1 Compilation

Exercice 1: Syntaxe du langage C

Lancez `emacs`, créez un buffer nommé `prog1.c`. Vérifiez que le mode `C` est utilisé par emacs dans ce buffer (barre d'état), sinon activez-le en tapant `M-x c-mode`. Tapez ensuite le programme suivant, et sauvegardez-le (clé `C-x C-s`).

```
void main() {
  printf("Hello world!\n");
}
```

Si votre programme n'apparaît que d'une couleur, activez la coloration syntaxique à l'aide de la commande emacs `M-x font-lock-mode`.

Exercice 2: Compilation

1. Pour compiler le programme précédent, tapez `M-x compile` et surveillez votre minibuffer. Remplacez la commande de compilation proposée par défaut (`make` ?) par :

   ```
   gcc prog1.c -Wall -o prog1
   ```

   et validez. Surveillez ce qui apparaît dans le buffer `*compilation*`. Corrigez votre programme et répétez les opérations sauvegarde/compilation jusqu'à ce que le message `Compilation finished at ...` apparaisse dans le buffer `*compilation*`.
   Dans un terminal, vérifiez (commande `ls -l`) qu'un fichier exécutable nommé `prog1` a bien été créé (dans le répertoire `TDc/td1/`). Lancez le avec la commande `./prog1`.

2. Modifiez votre programme en remplaçant `"Hello world!\n"` par `"Bonjour a tous !\n"`. Lancez l'exécutable `./prog1`, que s'est-il passé ? Que manque-t-il pour obtenir le résultat souhaité ?

il faut recompiler à chaque modification

3. Essayez de compiler vous-même votre programme dans le terminal, sans utiliser emacs, en tapant directement la commande de compilation ci-dessus (après vous être deplacé(e) dans le répertoire `TDc/td1` si nécessaire).

Exercice 3: Avertissements du compilateur

1. Compilez votre programme sous `emacs` et essayez la clé `C-x ‘` (backquote) plusieurs fois de suite, après que la compilation soit terminée.
2. Remplacez le mot `void` placé devant `main` par le mot `int`. Quelles sont les conséquences sur la compilation ? Et sur l'exécution ?
3. Ajoutez la ligne :

   ```
   return 0;
   ```

   juste avant la dernière accolade fermante. Quelles sont les conséquences sur la compilation ? Et sur l'exécution ?
4. Ajoutez la ligne :

   ```
   #include <stdio.h>
   ```

   en tête de votre programme. Quelles sont les conséquences sur la compilation ? Et sur l'exécution ?

Exercice 4: Erreurs de compilation

Créez un deuxième buffer `emacs` que vous nommez `prog2.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdo.h>

int man() {
  printf("Gasp !")

  return 0;
}
```

Compilez ce programme à l'aide de la commande :

```
gcc prog2.c -Wall -o prog2
```

Que constatez-vous ? Corrigez le programme jusqu'à ne plus avoir ni erreur, ni avertissements lors de la compilation.

2 Variables et types

Exercice 5: Un peu d'arithmétique

Créez un troisième buffer `emacs` que vous nommez `prog3.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  printf("division 2/3=%d\n", 2/3);
  printf("division 2.0/3.0=%f\n", 2.0/3.0);

  return 0;
}
```

%c → caractère
%d → entier
%f → flottant

Compilez ce programme, puis exécutez-le. Que constatez-vous ?

Exercice 6: Formats d'interprétation

Créez un quatrième buffer `emacs` que vous nommez `prog4.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  printf("division 2.0/3.0=%d\n", 2.0/3.0);

  return 0;
}
```

Compilez ce programme. Que constatez-vous ? Exécutez ce programme. Que constatez-vous ?

Exercice 7: Entiers signés/non signés

Créez un cinquième buffer `emacs` que vous nommez `prog5.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  unsigned int x = 4000000000U;

  printf("x=%u\n", x);
  printf("x=%d\n", x);

  return 0;
}
```

Compilez ce programme, puis exécutez-le. Que constatez-vous ?

Exercice 8: Interprétation des caractères

Créez un sixième buffer `emacs` que vous nommez `prog6.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  char c = 'A';

  printf("c=%c et c=%d\n", c, c);

  return 0;
}
```

Compilez ce programme, puis exécutez-le. Que constatez-vous ?

Exercice 9: Tailles des types de données

%lu → %u

Créez un septième buffer `emacs` que vous nommez `prog7.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  printf("taille de char = %lu\n", sizeof(char));
  printf("taille de unsigned char = %lu\n", sizeof(unsigned char));
  printf("taille de short = %lu\n", sizeof(short));
  printf("taille de unsigned short = %lu\n", sizeof(unsigned short));
  printf("taille de int = %lu\n", sizeof(int));
  printf("taille de unsigned int = %lu\n", sizeof(unsigned int));
  printf("taille de long = %lu\n", sizeof(long));
  printf("taille de unsigned long = %lu\n", sizeof(unsigned long));
  printf("taille de float = %lu\n", sizeof(float));
  printf("taille de double = %lu\n", sizeof(double));

  return 0;
}
```

Compilez et exécutez ce programme. Que signifient les valeurs affichées ?

Exercice 10: Interprétation des données

Créez un huitième buffer `emacs` que vous nommez `prog8.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  int i=2;
  float f=5.8;

  printf("i=%d, i=%f\n", i, i);
  printf("f=%f, f=%d\n", f, f);

  return 0;
}
```

Compilez ce programme. Que constatez-vous ? Exécutez ce programme que constatez-vous ?

Exercice 11: Compatibilité et forçage de types

Créez un neuvième buffer `emacs` que vous nommez `prog9.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  char c=70;
  int i=95;
  float f=124.932;

  printf("c=%d, c=%c, c=%f\n", c, c, (float)c);
  printf("i=%d, i=%c, i=%f\n", i, i, (float)i);
  printf("f=%d, f=%c, f=%f\n", (int)f, (int)f, f);

  return 0;
}
```

Compilez, puis exécutez ce programme. Que constatez-vous ?

Exercice 12: Débordements

Créez un dixième buffer `emacs` que vous nommez `prog10.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  char c = 128;
  unsigned int n = -1;

  printf("c=%d\n", c);
  printf("n=%u\n", n);
```

```
  return 0;
}
```

Compilez, puis exécutez ce programme. Que constatez-vous ?

Exercice 13: Compte à rebours

Créez un onzième buffer `emacs` que vous nommez `prog11.c`, et copiez-collez dans ce buffer le programme suivant :

```
#include <stdio.h>

int main() {
  unsigned char c;

  c = 10;
  while (c >= 0) {
    printf("%d\n", c);
    c--;
  }

  return 0;
}
```

Compilez, puis exécutez ce programme. Que constatez-vous ?