Yantra Technologies

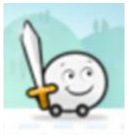
www.yantra-technologies.com

Le Langage C

Facile

Normal

Difficile



Professionnel

Expert

carole.grondein@yantra-technologies.com
david.palermo@yantra-technologies.com

https://wiki.waze.com/wiki/Your_Rank_and_Points

Programmation en C : Sommaire

1 - Généralités

2 - Compilation Gcc

3 - Le Langage C

4 - Bibliographie

V4,0 - Copyright : Yantra Technologies 2004-2015

1 - Généralités

1 - Historique

2 - Les styles de programmation

3 - Avantages du langage C

4 - Inconvénients du langage C

5 - Environnement de développement

V4,0 - Copyright : Yantra Technologies 2004-2015

1a- Historique

- Le langage C a été créé au début des années 1970 par Dennis Ritchie (Laboratoires Bell AT&T). Son but était de ré-écrire le système d'exploitation Unix en langage evolué.
  En 1978, le premier standard est publié ("The C Programming Language") par B. W. Kernighan et D.M. Ritchie.

- En 1983, l‘ "American National Standards Institute" (ANSI) définit le standard ANSI-C qui est une définition explicite et indépendante de la machine pour le langage C".

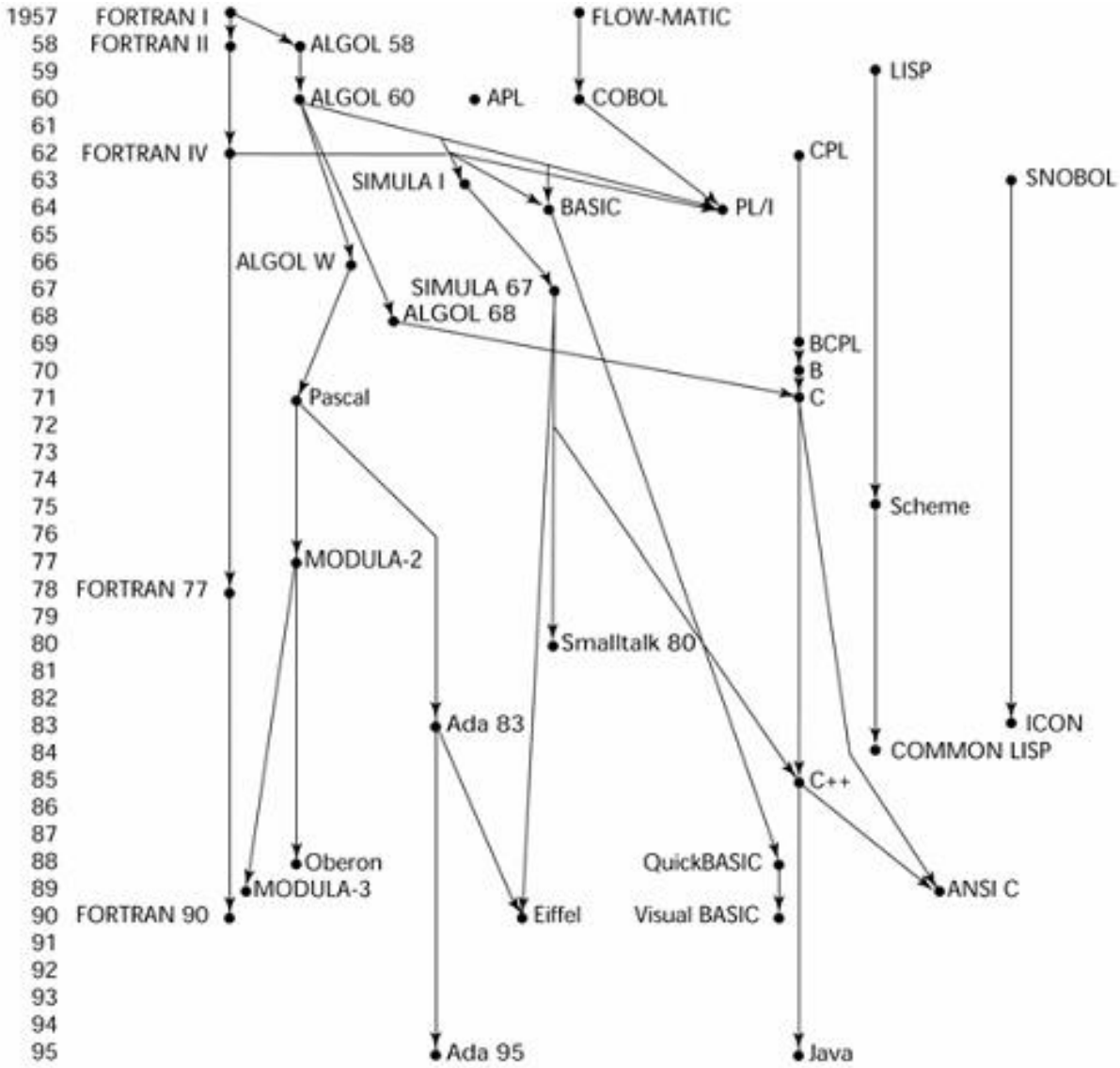
- Le langage C++ est né en 1983 et a pour but de développer un langage qui garderait les avantages de ANSI-C (portabilité, efficacité) et qui permettrait en plus la programmation orientée objet. Premier standard ANSI-C++ en 1998 corrigée en 2003,

- Nouvelle version 2011 nouvelle norme du C99 & C11

.

V4,0 - Copyright : Yantra Technologies 2004-2015

1b - Historique



V4,0 - Copyright : Yantra Technologies 2004-2015

2 - Les styles de programmation

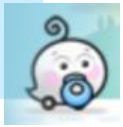
- **Fonctionnels : évaluation d 'expressions** : Lisp (1958), APL (récursivité) (1962), Caml (1985)

- **Procéduraux ou impératifs**: exécution d 'instructions
  Cobol (1959), *Fortran (1954), C (1970), Pascal (1970),*

- **Objet** : *Simula (1964-1967) , Smalltalk (1976), C++ (1980), Java (1995)*, *ADA 95 (1990), C# (2001)*
  - ensemble de composants autonomes (objets) qui disposent de moyens d 'interaction,
  - utilisation de classes,
  - échange de messages.

V4,0 - Copyright : Yantra Technologies 2004-2015

3 – Avantages du langage C



1. **Universel** : C n'est pas orienté vers un domaine d'applications spéciales
2. **Compact** : C est basé sur un noyau de fonctions et d'opérateurs limités qui permet la formulation d'expressions simples mais efficaces.
3. **Moderne** : C est un langage structuré, déclaratif et récursif. Il offre des structures de contrôle
4. **Près de la machine** : C offre des opérateurs qui sont très proches de ceux du langage machine (manipulations de bits, pointeurs…). C'est un atout essentiel pour la programmation des systèmes embarqués.
5. **Rapide** : C permet de développer des programmes concis et rapides.
6. **Indépendant de la machine** : C est un langage près de la machine (microprocesseur) mais il peut être utilisé sur n'importe quel système ayant un compilateur C. Au début C était surtout le langage des systèmes UNIX mais on le retrouve comme langage de développement aujourd'hui de tous les systèmes d'exploitation
7. **Portable** : En respectant le standard ANSI-C, il est possible d'utiliser (théoriquement ☺) le même programme sur tout autre système simplement en le recompilant. Il convient néanmoins de faire attention dans le cas d'un système embarqué qui n'inclut pas toujours un système d'exploitation…
8. **Extensible** : C peut être étendu et enrichi par l'utilisation de bibliothèques de fonctions achetées ou récupérées (logiciels libres…).

V4,0 - Copyright : Yantra Technologies 2004-2015

4 - Inconvénients du langage C

1. **Efficacité et compréhensibilité** : C autorise d'utiliser des expressions compactes et efficaces. Les programmes sources doivent rester compréhensibles pour nous-mêmes et pour les autres. ***Attention : La programmation efficace en langage C nécessite beaucoup d'expérience. Sans commentaires ou explications, les fichiers sources C peuvent devenir incompréhensibles et donc inutilisables (maintenance ?).***

2. **Portabilité et bibliothèques de fonctions :**
   - La portabilité est l'un des avantages les plus importants de C : en écrivant des programmes qui respectent le standard ANSI-C, nous pouvons les utiliser sur n'importe quelle machine possédant un compilateur ANSI-C.
   - Le nombre de fonctions standards ANSI-C (d'E/S) est limité. La portabilité est perdue si l'on utilise une fonction spécifique à la machine de développement.

3. **Discipline de programmation :**
   - C est un langage près de la machine donc dangereux bien que C soit un langage de programmation structurée.
   - C n'inclut pas de gestion automatique de la mémoire comme Java. ***La gestion mémoire en C est la cause de beaucoup de problèmes (***certains ont peut être eu une mauvaise expérience avec les *malloc()/free()*)…
   - L'instruction *goto* existe en C.

V4,0 - Copyright : Yantra Technologies 2004-2015

5 - Environnement de développement -> compilateur

MinGW ou Mingw32 (https://www.mingw-w64.org/) (Minimalist GNU for Windows) est une adaptation des logiciels de développement et de compilation du GNU (GCC - GNU Compiler Collection), à la plate-forme Win32.

le compilateur de Visual Studio (https://visualstudio.microsoft.com/fr/)

GNU Compiler Collection, abrégé en GCC, (https://gcc.gnu.org/) est un ensemble de compilateurs créés par le projet GNU. GCC est un logiciel libre capable de compiler divers langages de programmation, dont C, C++, Objective-C, Java, Ada, Fortran et Go.

Clang (https://clang.llvm.org/) est un compilateur pour les langages de programmation C, C++ et Objective-C. (altenative au GCC)

Intel C++/C Compiler (https://software.intel.com) également connu sous les sigles icc et icl) désigne une gamme de compilateurs C++ développés par Intel, disponibles pour les plates-formes Linux, Microsoft Windows et Mac OS X.

V4,0 - Copyright : Yantra Technologies 2004-2015
https://fr.wikipedia.org/wiki/Cat%C3%A9gorie:Compilateur_C

5 - Environnement de développement -> divers

Doxygen (https://www.doxygen.nl/index.html) : générateur de documentation sous licence libre capable de produire une documentation logicielle à partir du code source d'un programme.

Valgrind (https://www.valgrind.org/) : est un outil de programmation libre pour déboguer, effectuer du profilage de code et mettre en évidence des fuites mémoires.

KCachegrind (http://kcachegrind.sourceforge.net/html/Home.html) est un outil d'affichage des données de profilage, utilisé pour déterminer les segments prenant le plus de temps lors de l'exécution d'un programme.

Intel VTune Performance Analyzer (https://software.intel.com) Profiler est un outil d'analyse des performances pour les machines x86 exécutant les systèmes d'exploitation Linux ou Microsoft Windows. D

Intel Parallel Studio (https://software.intel.com) est un logiciel propriétaire développé par Intel. Il permet de développer des logiciels en C++ et Fortran. S

TotalView Debugger (https://totalview.io/free-trial) est un débogueur pour applications écrites en C/C++ ou Fortran.

Voir : http://c.developpez.com/compilateurs/

V4,0 - Copyright : Yantra Technologies 2004-2015

5 - Environnement de développement -> IDE

Eclipse CDT (C/C++ Development Tools) :

https://www.eclipse.org/cdt/

sous projet du projet "Eclipse Tools" dont le but est de fournir un environnement intégré de développement en C /C++ sous la forme de plug-ins pour Eclipse.

Code::Blocks (C/C++ et Fortran IDE)

https://www.codeblocks.org/

Environnement de développement intégré libre et multiplateforme C

Visual Studio Code

https://visualstudio.microsoft.com/fr/

IDE complet pour coder, déboguer, tester et déployer sur n'importe quelle plateforme.

https://www.codeur.com/blog/top-ide-environnement-de-developpement-c/
https://general.developpez.com/edi/
https://finobuzz.com/2017/07/04/les-10-ide-les-plus-connus-pour-programmer-en-c/

V4,0 - Copyright : Yantra Technologies 2004-2015

5 - Environnement de développement -> IDE

CodeLite

https://codelite.org/

IDE intégré pour les langages C/C++, ¨PHP, node js . Il est libre, open source et multiplateforme.

Clion

https://www.jetbrains.com/fr-fr/clion/

Clion est un multiplateforme C / C ++ IDE pour Linux, OS X et Windows intégré avec le système de construction CMak

C++ Builder

https://www.embarcadero.com/fr/products/cbuilder

Borland C++ Compiler est une version gratuite du compilateur C & C++ inclus dans C++ Builder. Il est fourni avec les en-têtes Win32 ainsi que make.

https://www.codeur.com/blog/top-ide-environnement-de-developpement-c/
https://general.developpez.com/edi/
https://finobuzz.com/2017/07/04/les-10-ide-les-plus-connus-pour-programmer-en-c/

V4,0 - Copyright : Yantra Technologies 2004-2015

5 - Environnement de développement -> IDE

NetBeans

https://netbeans.apache.org/kb/docs/cnd/

NetBeans est un IDE libre et gratuit, écrit en Java. Cet outil se compose d'une interface qui est munie d'une fonction de glisser-déposer et une liste de templates de projets préconstruits.

Anujta

http://anjuta.org/

Anjuta est un IDE (les fonctionnalités RAD intégrant Glade sont en cour de développement) pour Linux développé dans le cadre du projet Gnome et par conséquent, il est particulièrement intéressant pour les développements en C avec GTK+ (coloration syntaxique des mots clés de GTK+) mais il peut aussi être utilisé pour le développement en C++ avec gtkmm

https://www.codeur.com/blog/top-ide-environnement-de-developpement-c/
https://general.developpez.com/edi/
https://finobuzz.com/2017/07/04/les-10-ide-les-plus-connus-pour-programmer-en-c/

V4,0 - Copyright : Yantra Technologies 2004-2015

V4,0 - Copyright : Yantra Technologies 2004-2015

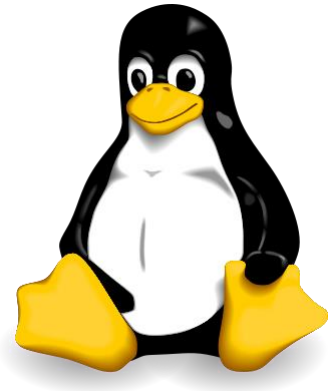
V4,0 - Copyright : Yantra Technologies 2004-2015

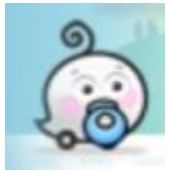
V4,0 - Copyright : Yantra Technologies 2004-2015

Yantra Technologies

www.yantra-technologies.com

Compilation Gcc





Facile

Normal

Difficile

Professionnel

Expert

carole.grondein@yantra-technologies.com
david.palermo@yantra-technologies.com

Sommaire

1 - Introduction

2 - Makefile

3 - Bibliographie

V4,0 - Copyright : Yantra Technologies 2004-2015

1.1 - Etapes de compilation

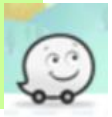
1.2 - Compilation

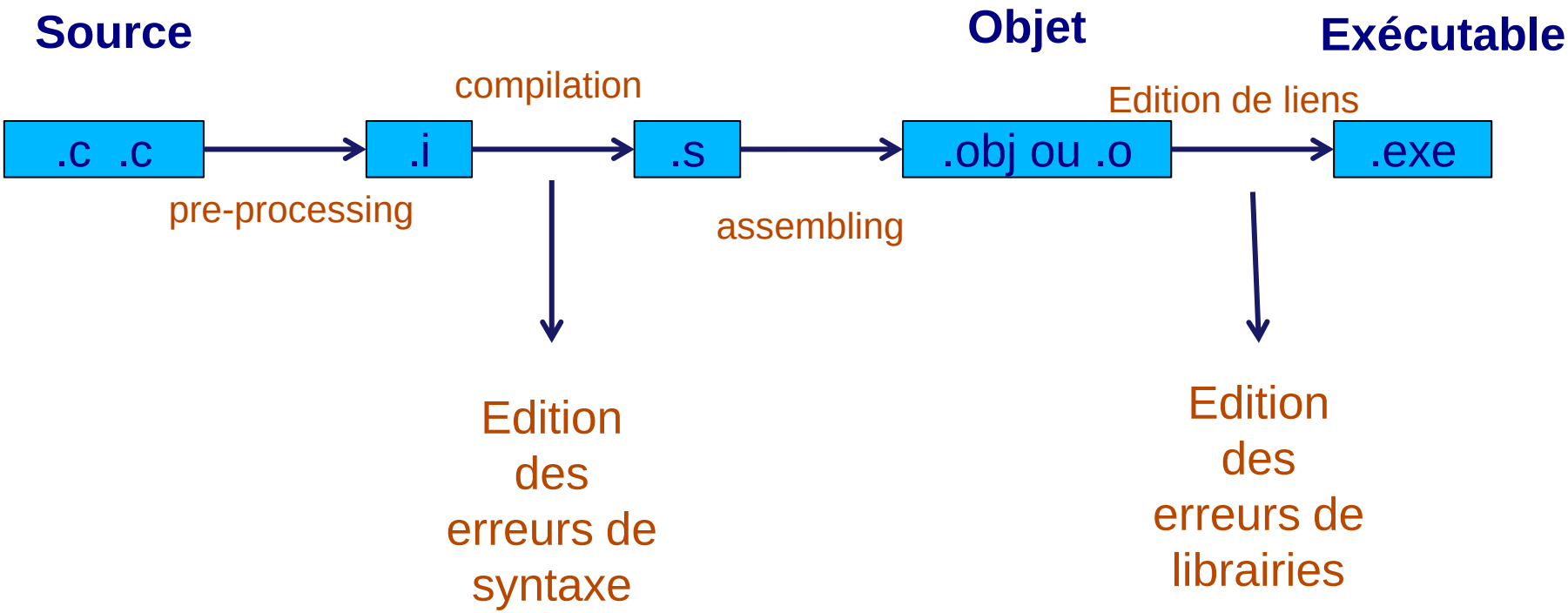
1.3 - Mon premier programme

1.4 - Ma première compilation

V4,0 - Copyright : Yantra Technologies 2004-2015

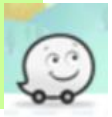
1.1 - Etapes de compilation





V4,0 - Copyright : Yantra Technologies 2004-2015

Préprocesseur

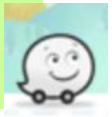


- suppression des commentaires (// et /* */)
- inclusion des fichiers .h ou .hh dans le fichier .c ou .c …. (#include)
- traitement des directives de compilation qui commencent par un caractère #.

gcc -E bonjour.c > bonjour.i ou gcc -E bonjour.c > bonjour.i

V4,0 - Copyright : Yantra Technologies 2004-2015

Compilation

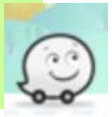


La Compilation le code ascii en code assembleur.

gcc -S bonjour.i ou gcc -S bonjour.i création d'un bonjour.o

V4,0 - Copyright : Yantra Technologies 2004-2015

Assembling



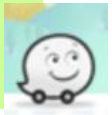
La Compilation le code ascii en code assembleur.

gcc bonjour.o ou gcc bonjour.o création d'un exécutable a.out

gcc –o bonjour bonjour.o ou gcc -o bonjour bonjour.o création d'un exécutable bonjour

V4,0 - Copyright : Yantra Technologies 2004-2015

Edition de lien

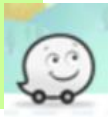


La Compilation le code ascii en code assembleur.

gcc –c bonjour.s ou gcc -c bonjour.s création de bonjour.o

V4,0 - Copyright : Yantra Technologies 2004-2015

Compilation simplifier



gcc –c bonjour.c

gcc –o bonjour bonjour.o création de l'exécutable bonjour

gcc –o bonjour bonjour.c création de l'exécutable bonjour

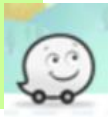
-c : Compile les fichiers sources, mais sans édition de liens. Le compilateur produit un fichier *objet* .o pour chaque fichier source.

-o *fichRes* : Donne le nom *fichRes* au fichier exécutable produit par la compilation et l'édition de liens. Quand cette option n'est pas employée, le nom d'exécutable par défaut est a.out.

-g : engendre de l'information pour le débogage, à utiliser avec un débogueur comme gdb (ou son interface utilisateur ddd).

V4,0 - Copyright : Yantra Technologies 2004-2015

Construction de bibliothèques



Bibliothèques statiques

En Unix ou Linux, pour réunir des fichiers objets, on utilisera typiquement la commande ar et ranlib, qui permet de créer des fichiers «archives» :

```
gcc –o F_1.o –c F_1.c
ar  -rv libmabib.a F_1.o F_2.o F_3.o
ranlib libmabib.a
```

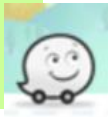
Bibliothèques dynamiques

En Unix ou Linux, pour réunir des fichiers objets, on utilisera l'option spécial –fPIC pour compiler les objets et –shared à l'édition de lien:

```
gcc –fPIC –o F_1.o –c F_1.c
gcc -shared -fPIC -o libmabib.so F_1.o F_2.o F_3.o
```


V4,0 - Copyright : Yantra Technologies 2004-2015

Utilisation de bibliothèques



Bibliothèques statiques

gcc –o mon_excutable main.c /chemin/libmabib.a

ou

gcc –static –o mon_excutable main.c –L/chemin **–l**mabib

Bibliothèques dynamiques

gcc –o mon_excutable main.c /chemin/libmabib.so

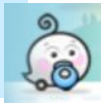
ou

gcc –o mon_excutable main.c –L/chemin –lmabib

Ne pas oublier de mettre à jour la variable LD_LIBRARY_PATH
LD_LIBRARY_PATH=$ LD_LIBRARY_PATH:/chemin

V4,0 - Copyright : Yantra Technologies 2004-2015

1.3.1 - Mon premier programme



```
#ifndef Bonjour_H
#define Bonjour_H

void Bonjour()  ;

#endif
```

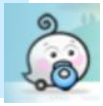
Bonjour.h

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

void Bonjour()  {
     printf("Bonjour ! \n");
}
```

Bonjour.c

V4,0 - Copyright : Yantra Technologies 2004-2015

1.3.1 - Mon premier programme

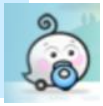


```
#include "Bonjour.h"

/* Mon premier programme  */

int main (int argc, char *argv[])
{
   Bonjour();
    return 0;
}
```

main.c

V4,0 - Copyright : Yantra Technologies 2004-2015

1.4 - Ma première compilation



gcc -c Bonjour.c

=> crée un fichier objet Bonjour.o

gcc -o Bonjour.exe main.c Bonjour.o

=> crée un exécutable Bonjour.exe

V4,0 - Copyright : Yantra Technologies 2004-2015

2 - Makefile

2.1 – Définition
2.2 – Mon premier Makefile
2.3 – Supprimer
2.4 - Définition Variable
2.5 - Variables internes
2.6 - Les règles d'inférence
2.7 - La cible .PHONY et .PRECIOUS
2.8 - Génération de la liste des fichiers objets
2.9 - Commandes silencieuses
2.10 - Les Makefiles conditionnels
2.11 - Création dépendance avec Gcc
2.12 - Création Librairie dynamic
2.13 - Option compilateur Gcc

V4,0 - Copyright : Yantra Technologies 2004-2015

2.1 - Définition

Un Makefile est un fichier constitué de plusieurs règles de la forme :

Cible **:** dépendance commandes
<tab> commande

Cible ou target
Chaque commande est précédée d'une tabulation.
Lors de l'utilisation d'un tel fichier via la commande make la première règle rencontrée, ou la règle dont le nom est spécifié, est évaluée.

L'évaluation d'une règle se fait en plusieurs étapes :

- Les dépendances sont analysées, si une dépendance est la cible d'une autre règle du Makefile, cette règle est à son tour évaluée.
- Lorsque l'ensemble des dépendances est analysé et si la cible ne correspond pas à un fichier existant ou si un fichier dépendance est plus récent que la règle, les différentes commandes sont exécutées.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.2 – Mon premier Makefile

```
Bonjour.exe: Bonjour.o main.o
    gcc -o Bonjour.exe Bonjour.o main.o

Bonjour.o: Bonjour.c Bonjour.h
    gcc -o Bonjour.o -c Bonjour.c -Wall –Wextra

main.o: main.c Bonjour.h
    gcc -o main.o -c main.c -Wall -Wextra
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.3 - Supprimer

```
all : Bonjour.exe
	echo "Executable crée"

Bonjour.exe: Bonjour.o main.o
	gcc -o Bonjour.exe Bonjour.o main.o

Bonjour.o: Bonjour.c Bonjour.h
	gcc -o Bonjour.o -c Bonjour.c -Wall –Wextra

main.o: main.c Bonjour.h
	gcc -o main.o -c main.c -Wall -Wextra

clean:
	 rm -rf *.o

mrproper: clean
	rm -rf Bonjour.exe
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

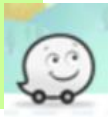
2.4 - Définition Variable

```
CXX=gcc
CXXFLAGS= -Wall –Wextra
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm

all : $(EXEC)
	echo "Executable crée"

Bonjour.exe: Bonjour.o main.o
	$(CXX) -o Bonjour.exe Bonjour.o main.o $(CXXLIBS)

Bonjour.o: Bonjour.c Bonjour.h
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) -o Bonjour.o -c Bonjour.c

main.o: main.c Bonjour.h
	$(CXX) $(CXXFLAGS) $(CXXINCLUDE) -o main.o -c main.c
clean:
	$(RM) -rf *.o

mrproper: clean
	$(RM) -rf $(EXEC)
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.5a - Variables internes



Il existe plusieurs variables internes au Makefile :

- $@ Le nom de la cible
- $< Le nom de la première dépendance
- $^ La liste des dépendances
- $? La liste des dépendances plus récentes que la cible
- $* Le nom du fichier sans suffixe

V4,0 - Copyright : Yantra Technologies 2004-2015

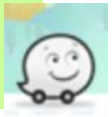
2.5b - Variables internes

```
CXX=gcc
CXXFLAGS= -Wall –Wextra
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm

all : $(EXEC)
	echo "Executable crée"

Bonjour.exe: Bonjour.o main.o
	$(CXX) -o $@ $^  $(CXXLIBS)

Bonjour.o: Bonjour.c Bonjour.h
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c $<
main.o: main.c Bonjour.h
	$(CXX) $(CXXFLAGS) $(CXXINCLUDE) -o $@ -c $<
clean:
	$(RM) -rf *.o

mrproper: clean
	$(RM) -rf $(EXEC)
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.6a - Les règles d'inférence



Makefile permet également de créer des règles génériques (par exemple construire un .o à partir d'un .c) qui se verront appelées par défaut.

Une telle règle se présente sous la forme suivante :

- %.o: %.c commandes

Il existe une autre notation plus ancienne de cette règle :

- .c.o: commandes

Il devient alors aisé de définir des règles par défaut pour générer nos différents fichiers.

V4,0 - Copyright : Yantra Technologies 2004-2015

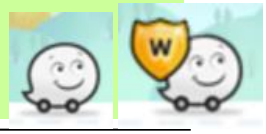
2.6b - Les règles d'inférence

```
CXX=gcc
CXXFLAGS= -Wall –Wextra
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm

all : $(EXEC)
	echo "Executable créé"

Bonjour: Bonjour.o main.o
	$(CXX) -o $@ $^ $(CXXLIBS)

%.o: %.c %.h
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c $<

clean:
	$(RM) -rf *.o

mrproper: clean
	$(RM) -rf $(EXEC)

main.o: Bonjour.h
Bonjour.o : Bonjour.h
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.7 - La cible .PHONY et .PRECIOUS



.PHONY Cible dont les dépendances seront systématiquement reconstruites.

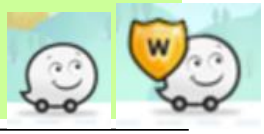
.PRECIOUS Cible dont les dépendances ne doivent pas être effacées automatiquement

```
CXX=gcc
CXXFLAGS= -Wall –Wextra
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm

all : $(EXEC)
	echo "Executable crée"

Bonjour.exe: Bonjour.o main.o
	$(CXX) -o $@ $^ $(CXXLIBS)

%.o: %.c %.h
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c $<

.PHONY: clean mrproper
.PRECIOUS : *.o

clean:
	$(RM) -rf *.o

mrproper: clean
	$(RM) -rf $(EXEC)

main.o: Bonjour.h
Bonjour.o : Bonjour.h
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.8a - Génération de la liste des fichiers objets



Plutôt que d'énumérer la liste des fichiers objets dans les dépendances de la règle de construction de notre exécutable, il est possible de la générer automatiquement à partir de la liste des fichiers sources. Pour cela nous rajoutons deux variables au Makefile :

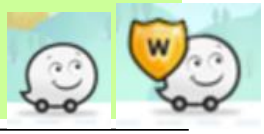
- SRC qui contient la liste des fichiers sources du projet.
- OBJ pour la liste des fichiers objets.

La variable OBJ est remplie à partir de SRC de la manière suivante :

OBJ= $(SRC:.c=.o) Pour chaque fichier .c contenu dans SRC nous ajoutons à OBJ un fichier de même nom mais portant l'extension .o.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.8b - Génération de la liste des fichiers objets

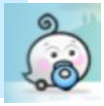


```
CXX=gcc
CXXFLAGS= -Wall –Wextra
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm
SRC= Bonjour.c main.c
OBJ= $(SRC:.c=.o)

all : $(EXEC)
	echo "Executable crée"

Bonjour.exe: $(OBJ)
	$(CXX) -o $@ $^  $(CXXLIBS)

%.o: %.c %.h
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c $<

.PHONY: clean mrproper
.PRECIOUS : $(OBJ)
clean:
	$(RM) -rf *.o

mrproper: clean
	$(RM) -rf $(EXEC)

main.o: Bonjour.h
Bonjour.o : Bonjour.h
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.9 - Commandes silencieuses



```
CXX=gcc
CXXFLAGS= -Wall –Wextra
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm
SRC= Bonjour.c main.c
OBJ= $(SRC:.c=.o)

all : $(EXEC)
	@echo "Executable crée"

Bonjour.exe: $(OBJ)
	@$(CXX) -o $@ $^  $(CXXLIBS)

%.o: %.c %.h
	@$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c $<

.PHONY: clean mrproper
.PRECIOUS : $(OBJ)
clean:
	@$(RM) -rf *.o

mrproper: clean
	@$(RM) -rf $(EXEC)

main.o: Bonjour.h
Bonjour.o : Bonjour.h
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.10 - Les Makefiles conditionnels

```
CXX=gcc
DEBUG=yes
ifeq ($(DEBUG),yes)
    CXXFLAGS= -Wall –Wextra -g
else
    CXXFLAGS= -Wall –Wextra
endif

EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm
SRC= Bonjour.c main.c
OBJ= $(SRC:.c=.o)

all : $(EXEC)
ifeq ($(DEBUG),yes)
    @echo " Executable crée en mode debug"
else
    @echo " Executable crée en mode release"
endif
```

```
Bonjour.exe: $(OBJ)
    @$(CXX) -o $@ $^  $(CXXLIBS)
%.o: %.c %.h
    @$(CXX) $(CXXFLAGS)
    $(CXXINCLUDE) –o $@ -c $<
.PHONY: clean mrproper
.PRECIOUS : $(OBJ)
clean:
    @$(RM) -rf *.o
mrproper: clean
    @$(RM) -rf $(EXEC)
main.o: Bonjour.h
Bonjour.o : Bonjour.h
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.11 - Création dépendance avec GCC

```
CXX=gcc
DEBUG=yes
ifeq ($(DEBUG),yes)
	CXXFLAGS= -Wall –Wextra -g
else
	CXXFLAGS= -Wall –Wextra
endif
EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS=
RM=rm
SRC= Bonjour.c main.c
OBJ= $(SRC:.c=.o)

all : $(EXEC)
ifeq ($(DEBUG),yes)
	@echo " Executable crée en mode debug"
else
	@echo " Executable crée en mode release"
endif

Bonjour.exe: $(OBJ)
	@$(CXX) -o $@ $^ $(CXXLIBS)
%.o: %.c %.h
	 @$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c $<

%.d: %.c
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) -MM -MP -MF"$@" -MT"$(@:%.d=%.o)" -c $<

.PHONY: clean mrproper
.PRECIOUS : $(OBJ)

clean:
	 @$(RM) -rf *.o
mrproper: clean
	@$(RM) -rf $(EXEC)
-include $(OBJ:.o=.d)
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.12 - Création Librairie static

```
CXX=gcc
DEBUG=yes
ifeq ($(DEBUG),yes)
	CXXFLAGS= -Wall –Wextra -g
else
	CXXFLAGS= -Wall –Wextra
endif

EXEC=Bonjour.exe
CXXINCLUDE=
CXXLIBS= -L.-lBonjour
RM=rm
SRC= Bonjour.c
OBJ= $(SRC:.c=.o)
MAIN=main.c
LIB=libBonjour.a

all : $(EXEC)
ifeq ($(DEBUG),yes)
	@echo " Executable crée en mode debug"
else
	@echo " Executable crée en mode release"
endif

Bonjour.exe: main.o $(LIB)
	@$(CXX) -o $@  $<  $(CXXLIBS)
```

```
libBonjour.a : $(OBJ)
	ar -cr $@ $^
	ranlib  $@

%.o: %.c %.h
	 @$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) –o $@ -c
	$<

%.d: %.c
	$(CXX) $(CXXFLAGS)  $(CXXINCLUDE) -MM -MP -
	MF"$@" -MT"$(@:%.d=%.o)" -c $<

.PHONY: clean mrproper
.PRECIOUS : $(OBJ)

clean:
	 @$(RM) -rf *.o
mrproper: clean
	@$(RM) -rf $(EXEC)

-include $(OBJ:.o=.d)
-include $(MAIN:.o=.d)
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.12 - Création Librairie dynamique

```
CXX=gcc
ifeq ($(DEBUG),yes)
CXXFLAGS= -Wall -Wextra -g
else
CXXFLAGS= -fpic -Wall -Wextra -pedantic -
        msse2 -mfpmath=sse
endif
EXEC=Bonjour-static Bonjour-dyn
CXXINCLUDE=-I .
CXXLIBS= -L. -lBonjour
RM=rm
SRC= Bonjour.c
OBJ= $(SRC:.c=.o)
MAIN=main.o
LIB=libBonjour.a libBonjour.so

all : $(EXEC)
ifeq ($(DEBUG),yes)
        echo " Executable crée en mode debug"
else
        echo " Executable crée en mode release"
endif
```

```
Bonjour-static: $(MAIN) $(LIB)
        $(CXX) -static -o $@  $<  $(CXXLIBS)
Bonjour-dyn: $(MAIN) $(LIB)
        $(CXX) -o $@  $< $(CXXLIBS)
libBonjour.a : $(OBJ)
        ar -cr $@ $^
        ranlib  $@
libBonjour.so : $(OBJ)
        $(CXX) -o $@ -shared $^
%.o: %.c %.h
         $(CXX) $(CXXFLAGS)  $(CXXINCLUDE) -o $@ -c $<
%.d: %.c
        $(CXX) $(CXXFLAGS)  $(CXXINCLUDE) -MM -MP -MF"$@" -
        MT"$(@:%.d=%.o)" -c $<
.PHONY: clean mrproper
.PRECIOUS : $(OBJ)
clean:
         @$(RM) -rf *.o *.d
mrproper: clean
        @$(RM) -rf $(EXEC) libBonjour.a
-include $(OBJ:.o=.d)
-include $(MAIN:.o=.d)
```

makefile

V4,0 - Copyright : Yantra Technologies 2004-2015

2.13a - Options compilateur Gcc

Option	Descriptif
-msse2	Active les extensions des processeurs Intel et AMD
-mfpmath=sse	Utilise l'extension SSE pour les calculs en nombre flottant
-mno-align-double **-malign-double**	Contrôle si GCC aligne les variables « double », « long double » et « long long » à des limites de deux ou de un mot. Aligner les variables « double » à des limites de deux mots produira du code tournant un peu plus rapidement sur un Pentium aux dépens d'une plus grosse consommation mémoire.
-m32	Force la compilation en 32 bits sur les machines 64 bits
-ansi :	Limite le langage à la norme ANSI C
-Wall :	affiche un maximum d'avertissements. Permet souvent d'éviter des bugs.
-Wextra	Active plus d'avertissements que -Wall (pour gcc < 4.0, l'option s'appelait "-W" tout court)
-fmessage-length=n	Essaye de formater les messages d'erreur de sorte qu'ils conviennent à des lignes d'environ *n* caractères. La valeur par défaut est de 72 caractères pour **gcc**, et 0 pour les autres frontaux supportés par GCC. Si *n* vaut zéro, aucun découpage en lignes ne sera effectué ; chaque message d'erreur apparaîtra sur une seule ligne.
-static	Crée un exécutable statique, ce qui accroit la portabilité.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.13b - Options compilateur Gcc

Option	Descriptif
-On	Indique le niveau d'optimisation (n peut prendre les valeurs allant de 0 à 3, ou « s » pour optimiser la taille des binaires)
-fabi-version=0	indique au compilateur de choisir une implémentation ABI particulière : • 0 : l'implémentation ABI la plus récente • 1 : l'implémentation ABI compatible gcc 3.2 • 2 : l'implémentation ABI la plus conforme
-fpic	Génère du code indépendant de son adresse de chargement (nécessaire pour les bibliothèques dynamiques, dont le chargement peut être réalisé par le système n'importe où en mémoire à priori
-march=cpu	Indique le type de processeur pour lequel le code doit être généré. Le code généré sera spécifique à ce processeur, et ne fonctionnera peut-être pas sur un autre modèle de la même famille. Cette option active automatiquement l'option -mcpu avec le même processeur.
-mtune=cpu	Fait en sorte que le code soit plus rapide sur architecture cpu tout en continuant à tourner sur i386.
-fbacktrace	Affiche lors d'une sortie en erreur du programme où l'erreur s'est produite
-fbounds-check	Permet la génération de contrôles durant l'exécution sur des indices de tableau et contre les valeurs minimales et maximales déclarées.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.13b - Options compilateur Gcc

Option	Descriptif
-fpermissive	autorise le code non conforme
-traditional	Essaie de supporter certains aspects des compilateurs C traditionnels. Pour les détails, voyez le manuel GNU C; la liste dupliquée ici a été supprimée afin de ne pas recevoir de réclamations quand elle n'est pas à jour. Mais notons une chose sur les programmes C++ uniquement (pas C). `-traditional' a un effet supplémentaire en C++ : l'affectation vers this est permise. Ceci a le même effet que `-fthis-is-variable
-Wunderflow	Produit un avertissement quand des expressions constantes numériques sont rencontrées, qui rapportent un dépassement de borne pendant la compilation
-shared	Produit un objet partagé qui peut être lié avec d'autres objets pour former un exécutable. Seul un petit nombre de systèmes supportent cette option.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.13c - Options compilateur Gcc

Option	Descriptif
-finit-integer	Sert à initialiser les variables de type integer
-finit-character	Sert à initialiser les variables de type character
-finit-logical	Sert à initialiser les variables de type logical
-finit-real	Sert à initialiser les variables de type real
-falign-commons	Sert à forcer l'alignement approprié de toutes les variables dans un bloc COMMUN en les alignant comme nécessaire. Sur certaines plates-formes cela est obligatoire, sur d'autres il augmente la performance à l'exécution
-fno-automatic	Rend toutes les variables locales static (SAVE)
-finit-local-zero	Sert à initialiser les variables locales à zéro

V4,0 - Copyright : Yantra Technologies 2004-2015

2.13d - Options compilateur Gcc

Option	Descriptif
-DSYMBOL[=valeur]	Cette option permet de définir des constantes ayant une valeur pour le préprocesseur. L'exemple précédent montre que le compilateur ajoute lui-même un grand nombre de ces constantes, permettant de caractériser le système sur lequel la compilation s'exécute (compilation conditionnelle).
-Ldirectory -llibrary	Ces options sont utilisées pour l'édition de liens. Elles permettent d'inclure des fichiers objets supplémentaires à ceux que l'on a compilé. Ces fichiers objets sont regroupés dans un fichier unique, nommé *librairie*. Une librairie est donc un ensemble de fichiers .o. Elle peut être liée statiquement ou dynamiquement à l'exécutable. L'option -Ldirectory indique le directory où le linker doit rechercher la librairie (au moment de l'édition des liens), et l'option -llibrary précise le nom de cette librairie. Selon les architectures, le fichier aura comme nom liblibrary.so pour une librairie dynamique ou library.a pour une librairie statique.
-IDirectory	Ajoute un répertoire pour la recherche des fichiers inclus.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.13e - Options compilateur Gcc

Option	Descriptif
-finline-functions	Active la fonction d'« inline »
-ffast-math et **-funsafe-math-optimizations**	Se permet de réorganiser les opérations mathématiques sur les nombres flottants. Par défaut, gcc se l'interdit car cette opération peut produire des erreurs de calcul (comprendre, sur les chiffres après les virgules, on peut perdre en précision).
-funroll-loops	Déroule les boucles
-pedantic	Encore plus restrictif que -ansi. En gros, si votre programme se compile avec -ansi, il se compile partout
-Werror	Les avertissements sont traités comme des erreurs

V4,0 - Copyright : Yantra Technologies 2004-2015

3 - Bibliographie

- Tutorial :
  - http://www.gnu.org/software/make/
  - http://gl.developpez.com/tutoriel/outil/makefile/
  - http://iihm.imag.fr/blanch/howtos/GNUMake.html
  - http://www.siteduzero.com/tutoriel-3-31992-compilez-sous-gnu-linux.html
  - http://www.linux-kheops.com/doc/man/manfr/man-ascii-0.9/man1/gcc.1.txt.html

V4,0 - Copyright : Yantra Technologies 2004-2015

V4,0 - Copyright : Yantra Technologies 2004-2015

V4,0 - Copyright : Yantra Technologies 2004-2015

Yantra Technologies

www.yantra-technologies.com

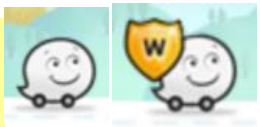
4 – Le Langage C

david.palermo@yantra-technologies.com

Références:

https://www.ltam.lu/Tutoriel-Ansi-C/
https://www.cplusplus.com/
http://public.iutenligne.net/informatique/algorithme-et-programmation/priou/LangageC/index.html

Sommaire



1 - Les commentaires
2 - Types prédéfinis
3 - Les variables
4 - Les opérateurs
5 - structures de contrôle
6 - Les pointeurs
7- Les tableaux et les chaînes de caractères
8 - Les fonctions
9 - Constantes & volatiles
10 - Saisie de nombres et Les chaînes de caractères
11 - Les types de variables complexes
13 - Les Directives de Précompilation
13 - Fichier
14 - Utilisation d'une bibliothèque
15 - Librairies standard
16 - Evolutions C99 et C11
17 - Supplément : Environnement Graphique : SDL

V4,0 - Copyright : Yantra Technologies 2004-2015

0 - Mots-clés du C++/C

alignas (depuis C++11)	char32_t (depuis C++11)	enum	namespace	return	try
alignof (depuis C++11)	class	explicit	new	short	typedef
and	compl	export	noexcept (depuis C++11)	signed	typeid
and_eq	const	extern	not	sizeof	typename
asm	constexpr (depuis C++11)	false	not_eq	static	union
auto (depuis C++11)	const_cast	float	nullptr (depuis C++11)	static_assert (depuis C++11)	unsigned
bitand	continue	for	operator	static_cast	using
bitor	decltype (depuis C++11)	friend	or	struct	virtual
bool	default	goto	or_eq	switch	void
break	delete	if	private	template	volatile
case	do	inline	protected	this	wchar_t
catch	double	int	public	thread_local (depuis C++11)	while
char	dynamic_cast	long	register	throw	xor
char16_t (depuis C++11)	else	mutable	reinterpret_cast	true	xor_eq
override (C++11)	final (C++11)				

V4,0 - Copyright : Yantra Technologies 2004-2015

1 - Les commentaires

Commentaire C :

/* Ceci est un commentaire C */

Depuis C99

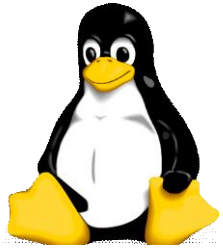
// ligne

V4,0 - Copyright : Yantra Technologies 2004-2015

Yantra Technologies

www.yantra-technologies.com

Outillage : Doxygen



carole.grondein@yantra-technologies.com
david.palermo@yantra-technologies.com

V4,0 - Copyright : Yantra Technologies 2004-2015

2 - Types prédéfinis

Type de données	Signification	Taille (en octets)	Plage de valeurs acceptées
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32 768 à 32 767
unsigned short int	Entier court non signé	2	0 à 65 535
int	Entier	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	-32 768 à 32 767 -2 147 483 648 à 2 147 483 647
unsigned int	Entier non signé	2 (sur processeur 16 bits) 4 (sur processeur 32 bits)	0 à 65 535 0 à 4 294 967 295
long int	Entier long	4	-2 147 483 648 à 2 147 483 647
unsigned long int	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$-3.4*10^{-38}$ à $3.4*10^{38}$
double	Flottant double	8	$-1.7*10^{-308}$ à $1.7*10^{308}$
long double	Flottant double long	10	$-3.4*10^{-4932}$ à $3.4*10^{4932}$
void	Type vide		

- les tableaux à une dimension, dont les indices sont spécifiés par des crochets ('[' et ']').
- les structures, unions et énumérations, les fonctions

Remarque :

- la taille des types peut dépendre des compilateurs,
- en hexadécimal, ils s'écrivent de la manière suivante : 127 -> 0x7f
- un bit est réservé pour le signe

V4,0 - Copyright : Yantra Technologies 2004-2015

2.1 - Types prédéfinis : <limits.h> & <float.h>

La norme nous fourni deux en-têtes définissant des macro constantes correspondant aux limites des types entiers et flottants : <limits.h> et <float.h>.

V4,0 - Copyright : Yantra Technologies 2004-2015

2.1 - Types prédéfinis : <limits.h> & <float.h>

name	expresses	possible value*
CHAR_BIT	Number of bits in a char object (byte)	8 or greater*
SCHAR_MIN	Minimum value for an object of type signed char	-127 (-2^7+1) or less*
SCHAR_MAX	Maximum value for an object of type signed char	127 (2^7-1) or greater*
UCHAR_MAX	Maximum value for an object of type unsigned char	255 (2^8-1) or greater*
CHAR_MIN	Minimum value for an object of type char	either SCHAR_MIN or 0
CHAR_MAX	Maximum value for an object of type char	either SCHAR_MAX or UCHAR_MAX
MB_LEN_MAX	Maximum number of bytes in a multibyte character, for any locale	1 or greater*
SHRT_MIN	Minimum value for an object of type short int	-32767 ($-2^{15}+1$) or less*
SHRT_MAX	Maximum value for an object of type short int	32767 ($2^{15}-1$) or greater*
USHRT_MAX	Maximum value for an object of type unsigned short int	65535 ($2^{16}-1$) or greater*
INT_MIN	Minimum value for an object of type int	-32767 ($-2^{15}+1$) or less*
INT_MAX	Maximum value for an object of type int	32767 ($2^{15}-1$) or greater*
UINT_MAX	Maximum value for an object of type unsigned int	65535 ($2^{16}-1$) or greater*
LONG_MIN	Minimum value for an object of type long int	-2147483647 ($-2^{31}+1$) or less*
LONG_MAX	Maximum value for an object of type long int	2147483647 ($2^{31}-1$) or greater*
ULONG_MAX	Maximum value for an object of type unsigned long int	4294967295 ($2^{32}-1$) or greater*
LLONG_MIN	Minimum value for an object of type long long int	-9223372036854775807 ($-2^{63}+1$) or less*
LLONG_MAX	Maximum value for an object of type long long int	9223372036854775807 ($2^{63}-1$) or greater*
ULLONG_MAX	Maximum value for an object of type unsigned long long int	18446744073709551615 ($2^{64}-1$) or greater*

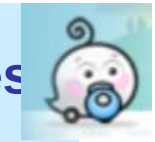
V4,0 - Copyright : Yantra Technologies 2004-2015

2.1 - Types prédéfinis : <limits.h> & <float.h>

name	value	stands for	expresses
FLT_RADIX	2 or greater	RADIX	Base for all floating-point types (float, double and long double).
FLT_MANT_DIG DBL_MANT_DIG LDBL_MANT_DIG		MANTissa DIGits	Precision of *significand*, i.e. the number of digits that conform the *significand*.
FLT_DIG DBL_DIG LDBL_DIG	6 or greater 10 or greater 10 or greater	DIGits	Number of **decimal digits** that can be rounded into a floating-point and back without change in the number of decimal digits.
FLT_MIN_EXP DBL_MIN_EXP LDBL_MIN_EXP		MINimum EXPonent	Minimum negative integer value for the *exponent* that generates a normalized floating-point number.
FLT_MIN_10_EXP DBL_MIN_10_EXP LDBL_MIN_10_EXP	-37 or smaller -37 or smaller -37 or smaller	MINimum base-10 EXPonent	Minimum negative integer value for the *exponent* of a base-10 expression that would generate a normalized floating-point number.
FLT_MAX_EXP DBL_MAX_EXP LDBL_MAX_EXP		MAXimum EXPonent	Maximum integer value for the *exponent* that generates a normalized floating-point number.
FLT_MAX_10_EXP DBL_MAX_10_EXP LDBL_MAX_10_EXP	37 or greater 37 or greater 37 or greater	MAXimum base-10 EXPonent	Maximum integer value for the *exponent* of a base-10 expression that would generate a normalized floating-point number.
FLT_MAX DBL_MAX LDBL_MAX	1E+37 or greater 1E+37 or greater 1E+37 or greater	MAXimum	Maximum finite representable floating-point number.
FLT_EPSILON DBL_EPSILON LDBL_EPSILON	1E-5 or smaller 1E-9 or smaller 1E-9 or smaller	EPSILON	Difference between 1 and the least value greater than 1 that is representable.
FLT_MIN DBL_MIN LDBL_MIN	1E-37 or smaller 1E-37 or smaller 1E-37 or smaller	MINimum	Minimum representable positive floating-point number.
FLT_ROUNDS		ROUND	Rounding behavior. Possible values: -1 undetermined 0 toward zero 1 to nearest 2 toward positive infinity 3 toward negative infinity Applies to all floating-point types (float, double and long double).
FLT_EVAL_METHOD		EVALuation METHOD	Properties of the evaluation format. Possible values: -1 undetermined 0 evaluate just to the range and precision of the type 1 evaluate float and double as double, and long double as long double. 2 evaluate all as long double Other negative values indicate an implementation-defined behavior. Applies to all floating-point types (float, double and long double).
DECIMAL_DIG		DECIMAL DIGits	Number of decimal digits that can be rounded into a floating-point type and back again to the same decimal digits, without loss in precision.

4,0 - Copyright : Yantra Technologies 2004-2015

2.2 - Types prédéfinis : Quelques constantes caractères

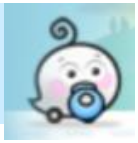


le type char est un type entier particulier

CARACTERE	VALEUR (Code ASCII)	NOM ASCII	
'\n'	interligne	0x0a	LF
'\t'	tabulation horizontale	0x09	HT
'\v'	tabulation verticale	0x0b	VT
'\r'	retour charriot	0x0d	CR
'\f'	saut de page	0x0c	FF
'\\'	backslash	0x5c	\
'\''	cote	0x2c	'
'\"'	guillemets	0x22	"
'\a'	signal alerte		
'\b'	retour arriere		

V4,0 - Copyright : Yantra Technologies 2004-2015

2.2 - Types prédéfinis : Déclaration type réel



Forme :

<+/-> mantisse * [E,e] <exposant>

Composés :

- d'un signe : + ou -
- d'une mantisse : chiffre positif à 1 chiffre avant la virgule
- d'un exposant : entier relatif

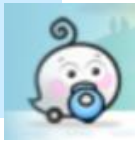
3 principaux types : float, double, long float

Exemples :

- double d = -1.E5;

V4,0 - Copyright : Yantra Technologies 2004-2015

3.1 – Les variables -> Déclarer une variable

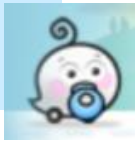


- Forme
  - type nomdevariable;

- Exemples
  - int i; /* déclaration de la variable i */
  - char c; /* déclaration du caractère c */

V4,0 - Copyright : Yantra Technologies 2004-2015

3.2 – Les variables -> Initialiser une variable

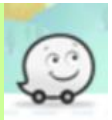


C permet l'initialisation des variables dans la zone de déclaration

Affecter		Initialiser
char c; c='A';	abouti au même résultat que	char c='A';
int i; i=50;	abouti au même résultat que	int i=50;
const int i ; Erreur *i=50*		*const int i = 50;*

V4,0 - Copyright : Yantra Technologies 2004-2015

4 – Les opérateurs



4.1 - Les opérateurs : arithmétiques

4.2 - Les opérateurs : binaires

4.3 - Les opérateurs : conditionnels

4.4 - Les opérateurs : simplifiés

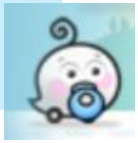
4.5 - Les opérateurs : logiques

4.6 - Opérateur sizeof()

4.7 - Opérateur de cast ()

V4,0 - Copyright : Yantra Technologies 2004-2015

4.1 - Les opérateurs : arithmétiques



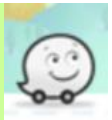
- Sur les réels : + - / *
- Sur les entiers : + - / (quotient de la division), % (reste de la division)

l'affectation se fait grâce au signe '='

Hiérarchie habituelle (* et / prioritaires par rapport aux + et -)

V4,0 - Copyright : Yantra Technologies 2004-2015

4.2a - Les opérateurs : binaires



- &, et (and)

Table de vérité AND

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- |, ou, (OR)

Table de vérité OR

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

- ^, ou exclusif (XOR)

Table de vérité XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

V4,0 - Copyright : Yantra Technologies 2004-2015

4.2b - Les opérateurs : binaires

- ~, complément à 1

Table de vérité NOT

A	NOT A
0	1
1	0

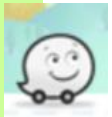
- <<, décalage à gauche
- >>, décalage à droite

Exemple : p = n « 3;

p égal n décalé de 3 bits sur la gauche

V4,0 - Copyright : Yantra Technologies 2004-2015

4.3 - Les opérateurs : conditionnels



expr1 ? expr2 : expr3

si (expr1) alors expr2 sinon expr3

Exemple : fonction min

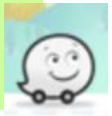
```
if (y<z)  x=y; else x=z;
```

Equivalent à

```
x = ( y<z) ? y : z ;
```


V4,0 - Copyright : Yantra Technologies 2004-2015

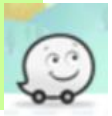
4.4 - Les opérateurs : simplifiés



- i=i+1 peut s'écrire i++; (++i;)
- i=i-1 peut s'écrire i--; (--i;)
- a=a+b peut s'écrire a+=b;
- a=a-b peut s'écrire a-=b;
- a=a&b peut s'écrire a&=b;

V4,0 - Copyright : Yantra Technologies 2004-2015

4.5 - Les opérateurs : logiques



4.5.1 - Quelle utilisation ?

4.5.2 - Opérateurs classiques

4.5.3 - Les expressions

4.5.4 - Et - Ou - !

V4,0 - Copyright : Yantra Technologies 2004-2015

4.5.1 - Quelle utilisation ?

- Tous les tests :
  - si (expression vraie) ...
  - tant que (expression vraie) ...

- Valeurs booléennes
  - Vrai (1)
  - Faux (0)

V4,0 - Copyright : Yantra Technologies 2004-2015

4.5.2 - Opérateurs classiques

- Le 'ET'

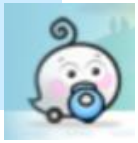
A	B	ET
0	0	0
0	1	0
1	0	0
1	1	1

- Le 'OU'

A	B	OU
0	0	0
0	1	1
1	0	1
1	1	1

V4,0 - Copyright : Yantra Technologies 2004-2015

4.5.3 - Les expressions



- Egalité

  a == b

- Inégalité

  a != b

- Relations d'ordre

  a < b a <= b a > b a >= b

- Expression

  a est vrai si a est différent de 0

V4,0 - Copyright : Yantra Technologies 2004-2015

4.5.4 - Et - Ou - !

- Et Logique
  - expression1 && expression2
  - si expression1 et expression2 sont vraies
- Ou Logique
  - expression1 || expression2
  - si expression1 ou expression2 est vraie
- ! (Non)
  - !expression
  - si expression est fausse

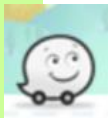
V4,0 - Copyright : Yantra Technologies 2004-2015

4.5.5 - Priorité des opérateurs

priorité	opérateur	parité	associativité	description
1	( )		gauche vers la droite (GD)	parenthésage
2	() [] . ->		GD	appel de fonction, index de tableau, membre de structure, pointe sur membre de structure
3	!	unaire	droite vers la gauche (DG)	négation booléenne
4	~	unaire	DG	négation binaire
5	++ --	unaire	DG	incrémentation et décrémentation
6	-	unaire	DG	opposé
7	(type)	unaire	DG	opérateur de transtypage (cast)
8	*	unaire	DG	opérateur de déréférençage
9	&	unaire	DG	opérateur de référençage
10	sizeof	unaire	DG	fournit la taille en nombre de "char" de l'expression (souvent en octet mais pas toujours, mais sizeof(char) == 1 par définition, voir Caractères)
11	* / %	binaire	GD	multiplication, division, modulo (reste de la division)
12	+ -	binaire	GD	addition, soustraction

V4,0 - Copyright : Yantra Technologies 2004-2015

4.5.5 - Priorité des opérateurs



priorité	opérateur	parité	associativité	description
13	>> <<	binaire	GD	décalages de bits
14	> >= < <=	binaire	GD	comparaisons
15	== !=	binaire	GD	égalité/différence
16	&	binaire	GD	et binaire
17	^	binaire	GD	ou exclusif binaire
16	|	binaire	GD	ou inclusif binaire
19	&&	binaire	GD	et logique avec séquencement
20	||	binaire	GD	ou logique avec séquencement
21	? :	ternaire	DG	si...alors...sinon
22	= += -= *= /= %= ^= &= |= >>= <<=	binaire	DG	affectation
23	,	binaire	GD	séquencement

V4,0 - Copyright : Yantra Technologies 2004-2015

std::size_t sizeof(type)

Operateur qui rends la taille de l'objet ou de type passé en paramètre

```
#include <iostream>
struct Empty {};
struct Bit {unsigned bit:1; };
int main()
{
    Empty e;
    Bit b;
    std::cout << "size of empty class: "     << sizeof e        << '\n'
              << "size of pointer : "        << sizeof &e       << '\n'
//            << "size of function: "        << sizeof(void())  << '\n'  // compile error
//            << "size of incomplete type: " << sizeof(int[])   << '\n'  // compile error
//            << "size of bit field: "       << sizeof b.bit    << '\n'  // compile error
              << "size of array of 10 int: " << sizeof(int[10]) << '\n';
}
```

Résultat :

```
size of empty class: 1
size of pointer : 8
size of array of 10 int: 40
```

https://fr.cppreference.com/w/cpp/language/sizeof

V4,0 - Copyright : Yantra Technologies 2004-2015

4.7 - Opérateur de cast ()

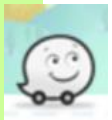
L'opérateur de conversion de type s'appelle un cast.

Un cast de type fournit une méthode pour la conversion explicite du type d'un objet dans une situation spécifique.

```
#include <iostream>

using namespace std;

int main()
{
    double x = 3.1;
    int i;
    cout << "x = " << x << endl;
    i = (int)x;   // assign i the integer part of x
    cout << "i = " << i << endl;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5 - Les structures de contrôle



5.1 - Si … alors … sinon … => if () {} else {}

5.2 - Au cas … ou faire … => switch () { case }

5.3 - Tant que … faire … => while () {}

5.4 - Répéter … tant que …=> do {} while ()

5.5 - Boucle Pour … faire … => for () {}

5.6 - Saut => goto

5.7- Rupture de séquence => break, continue, return

V4,0 - Copyright : Yantra Technologies 2004-2015

5.1 - si … alors … sinon … => if () {} else {}

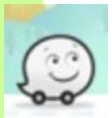
Si (expression conditionnelle vraie)
alors { instructions 1 }
sinon { instructions 2 }

```
if (expression) {
   instructions 1;
}
else {
   instructions 2;
}

if (expression) {
   instructions 1;    /* bloc else
    optionnel */
}
```

```
if (expression)
  instruction;    /* bloc d'1
    instruction */

if (expression) instruction1;
else instruction2;
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.1 - Exemple



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

int main(int argc, char *argv[]) {
    char quit = '\0';
    printf(" Bonjour ! \n");
    printf(" Appuyer sur la touche q pour quitter \n");
    quit=getchar();
    if ( quit == 'q' ) return 0;

    if ( quit == 'Q' ) {
     printf(" Appuyer sur la touche q et pas Q pour quitter \n");
     quit=getchar();
    }
    else
    {
     printf("  Appuyer sur la touche q pour quitter \n");
     quit=getchar();
    }

    if ( quit == 'q' ) return 0;
    else printf("Dernier essai, Appuyer sur la touche q pour quitter \n");

    quit=getchar();
    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.2 - Au cas … ou faire …=> switch () { case }

Au cas ou la variable vaut :

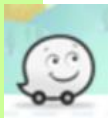
valeur 1 : faire … ;

valeur 2 : faire … ; etc.

```
switch (variable de type char ou int)  {
    case valeur 1 : …;
            …;
            break;
    case valeur 2 : …;
            …;
            break;
    default :          …;
            …;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.2 - Exemple



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

int main(int argc, char *argv[])
{
   char quit = '\0';
   printf ( " Bonjour ! \nAppuyer sur la touche q pour quitter \n");
   quit=getchar();
   switch ( quit )
   {
      case 'q' :
            return 0; break;
      case 'Q' :
            printf ( " Appuyer sur la touche q et pas %c pour quitter \n",quit);
          quit=getchar();
            break;
      case 'r':case 'R':case 'S' : case 'T' :
            printf ( " Appuyer sur la touche q et pas %c pour quitter \n",quit);
          quit=getchar();
            break;
      default :
            printf ( " Appuyer sur la touche q et pas %c pour quitter \n",quit);
            quit=getchar();
            break;
   }
   if ( quit == 'q' ) return 0;
   else printf ( "Dernier essai, Appuyer sur la touche q pour quitter \n");
   quit=getchar();
   return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.3 - Tant que … faire => while () {}

Tant que (expression vraie) faire
{bloc d ’instructions}

```
while (expression vraie)  {
    bloc d ’instructions }
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.3 - Exemple

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

int main(int argc, char *argv[])
{
    char quit = '\0';
    while ( quit != 'q' )
    {
      printf ( " Bonjour ! \n Appuyer sur la touche q pour quitter \n");
      quit=getchar();
    }
    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.4 - Répéter … tant que …=> do {} while ()

Répéter {bloc d ’instructions}
tant que (expression vraie)

```
do {bloc d ’instructions} while
(expression vraie);
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.4 - Exemple

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

int main(int argc, char *argv[])
{
    char quit = '\0';
    do
    {
      printf ( " Bonjour ! \n Appuyer sur la touche q pour quitter \n");
      quit=getchar();
    }
    while ( quit != 'q' );

    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

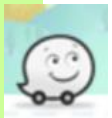
5.5 - Boucle Pour … faire … => for () {}

Pour (initialisation;condition;modification) faire {
bloc d ’instructions}

```
for (initialisation;condition;modification)  {
    bloc d ’instructions}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

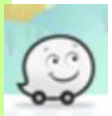
5.5 - Exemple



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

int main(int argc, char *argv[])
{
    char quit = '\0';
    int i;
    printf ( " Bonjour ! \n Appuyer sur la touche q pour quitter \n");
    quit=getchar();
    for ( ; quit!='q'; )
    {
        for (i = 0 ; i < 5; i++) printf ( "Bonjour ! \n");
        printf ( " Appuyer sur la touche q pour quitter => ");
        scanf("%c",&quit);getchar();
    }
    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.6 - Saut => goto



goto étiquette ;
étiquette :

```
goto etiquette ;
etiquette :
```

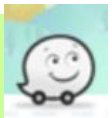

V4,0 - Copyright : Yantra Technologies 2004-2015

5.6 - Exemple

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

int main(int argc, char *argv[])
{
    char quit = '\0';

    reprise :

    printf ( " Bonjour ! \n Appuyer sur la touche q pour quitter \n");
     scanf("%c",&quit);getchar();
     if ( quit!='q' ) goto reprise;

    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

5.7 - Rupture de séquence => break, continue, return



- ***Return[valeur]*** : permet de quitter immédiatement la fonction en cours.

- ***break*** : permet de passer à l'instruction suivant l'instruction *while*, *do*, *for* ou *switch* la plus imbriquée.

- ***continue*** : saute directement à la dernière ligne de l'instruction *while*, *do* ou *for* la plus imbriquée.

V4,0 - Copyright : Yantra Technologies 2004-2015

5.7 – Exemple

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

int main(int argc, char *argv[])
{
   char quit = '\0';
     int i;
   while ( quit!='q' )
   {
      for (i=0;i< 5;i++)
      {
            printf ( " Bonjour ! : num => %i \n",i);
            printf ( " Appuyer sur la touche 'o' pour quitter la boucle\n");
            scanf("%c",&quit);getchar();
            if ( quit=='o' ) break;
      }
   printf ( " Appuyer sur la touche 'r' pour recommencer la boucle\n");
   printf ( " Appuyer sur la touche 'q' pour quitter \n");
   scanf("%c",&quit);getchar();

   if ( quit=='r' ) continue;
   if ( quit!='q' ) return 0;
   }
   return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6 - Les pointeurs

6.1 - Les Pointeurs
6.2 - Opérateur
6.3 - Allocation dynamique de mémoire
6.4 - Arithmétique des pointeurs
6.5 - Opérateur de Déréférencement et d'indirection
6.6 - Allocation dynamique
6.7 - Libération de la mémoire
6.8 - Allocation dynamique en C
6.9 - Attention
6.10 - Pointeurs constants ou volatiles
6.11 - Fonction scanf
6.12 - Passage de paramètres par variable ou par valeur
6.13 - Pointeur de fonctions
6.14 - Conclusion

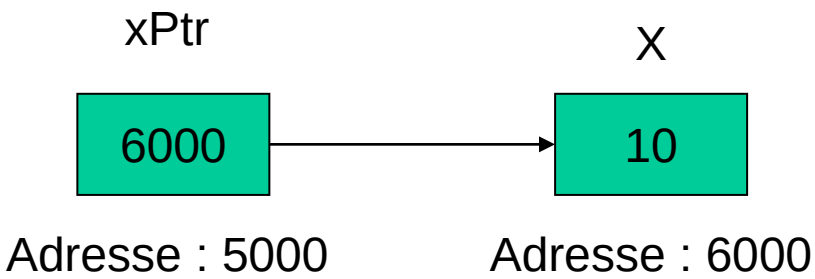
V4,0 - Copyright : Yantra Technologies 2004-2015

6.1a - Définition Pointeur

- Tout emplacement mémoire a une adresse
- Un pointeur est une variable qui contient une adresse mémoire
- On dit que le pointeur pointe sur cette adresse

Déclaration pointeur
pointeur = type *** p**
Récupération valeur pointée
*** p**

```
int main ()
{
  int x = 10;
  int * xPtr;  /* xPtr pointeur pointant sur type int */
  xPtr = &x;
}
```




```
5000 6000 10
6000 10
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.1b - Définition Adresse

Notion :

- Un objet manipulé est stocké dans sa mémoire.
- Une mémoire est constituée d'une série de « cases », ou sont stockées des valeurs (variables ou instructions)
- Pour accéder au contenu de la case mémoire où l'objet est enregistré, il faut connaître l'emplacement en mémoire de l'objet à manipuler.

Cet emplacement est appelé l'adresse de la case mémoire, ou l'adresse de la variable ou l'adresse de la fonction.

Toute case mémoire a une adresse unique.

int x = 10

X

10

Adresse : 6000

<u>Notation adresse</u>

adresse x = **&x**

V4,0 - Copyright : Yantra Technologies 2004-2015

6.2 - Opérateur &

- Retourne l ’adresse d’une variable en mémoire
- cf scanf

  – Exemple :

```
int i=8;
int* iptr = &i;
printf(“Voici i et son adresse %d , %p“,i,&i);
printf(“Voici iptr et son adresse %p , %p, %d “,iptr,&iptr,*iptr);
*iptr=5;
printf(“Voici i et son adresse %d , %p“,i,&i);
printf(“Voici iptr et son adresse %p , %p, %d “,iptr,&iptr,*iptr);
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.3 - Allocation dynamique de mémoire

L'allocation dynamique de mémoire permet de réserver de la mémoire (appelée encore *allocation*) pendant l'exécution d'un programme.

La quantité de mémoire utilisée par le programme est variable.

V4,0 - Copyright : Yantra Technologies 2004-2015

6.4a - Arithmétique des pointeurs

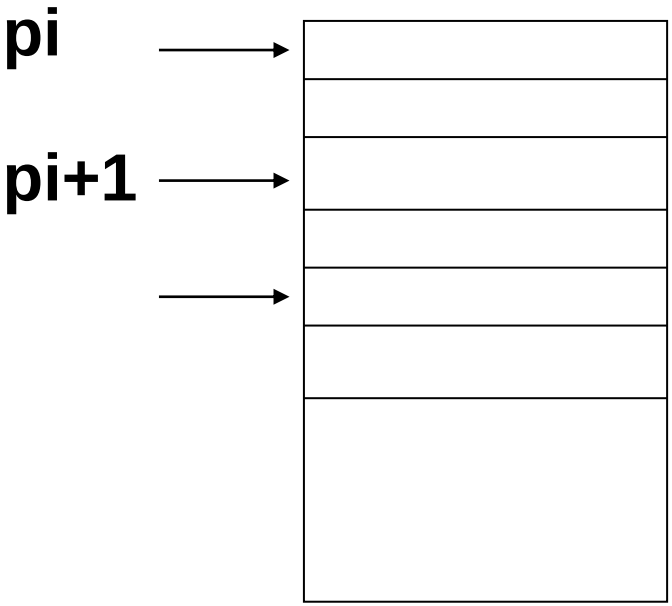
- Déplacement du pointeur dans l'espace mémoire
- Déplacement multiple du type de variable pointée

  – Exemples :

```
int i = 8;
int *pi = NULL;
pi = &i ;
*pi=12;
/*  *(pi+1)=3; */
printf(''%d %d %p %p %p \n'', i, *pi, &i, pi , &pi);
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.4b - Arithmétique des pointeurs




```
int*pi;
*pi=0;
```

1 entier = 2 octets

V4,0 - Copyright : Yantra Technologies 2004-2015

6.5 - Opérateur de Déréférencement et d'indirection

```
int i = 0;
int *pi = 0;
pi = &i;
*pi = *pi+1;
```

Attention *pi+1 != *(pi+1)

V4,0 - Copyright : Yantra Technologies 2004-2015

6.6 - Allocation dynamique

- Réservation obligatoire pour pérennité
- Allocation par la fonction malloc
- Prototype dans <stdlib.h>
- Exemples :

```
int i=0;
char * pc=(char*)malloc(10);
int* pj= 0 ;
/*pi=(int*)malloc(16);*/
int* pi = (int*) malloc(4*sizeof(int));
for (i=0;i < 4 ;i++) {
        pi[i]=i; /* *(pi+i)=i;*/
        printf("%d \n",pi[i]);
}
pj=(int*)malloc(sizeof(int));
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.7 - Libération de la mémoire

- Fonction free(pointeur)
- Evite l'encombrement mémoire par la destruction des pointeurs

- Exemples :

```
free(pi);
free(pc);
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.8 - Allocation dynamique en C

Allocation dynamique : malloc

- signature de la fonction :
  - void* malloc(unsigned int) ;
- Prototype dans alloc.h
- Exemples :
  - char* pc;
  - pc=(char*) malloc(10);
  - int * pi;
  - pi=(int*) malloc(sizeof(int)*16);

Libération mémoire : free

- Libération par la fonction
  - free(pointeur)
- Évite l'encombrement mémoire par la destruction des pointeurs
- Exemples :
  - free(pi);
  - free(pc);

V4,0 - Copyright : Yantra Technologies 2004-2015

6.9 - Attention

- Ne pas affecter directement une valeur dans un pointeur !
  - Exemple : **char *pc;**

    pc=0xfffe;

V4,0 - Copyright : Yantra Technologies 2004-2015

6.10 - Pointeurs constants ou volatiles

```
int const *pi;
const int *pi = (int*)malloc(sizeof(int));  /* pi est un pointeur sur un entier
                                               constant. */
/* pi = 2; erreur */
pi = (int*)malloc(sizeof(int));

int j;
int * const pj=&j; /* pi est  un pointeur constant sur un entier non constant */

*pj = 2;
/* pj = (int*)malloc(sizeof(int));  erreur*/

const int * const pc = &j;

const int *pi[10]; /* pi est un tableau de 10 pointeurs d'entiers constants */
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.11 - Fonction scanf

- Utilisation : scanf(“%d“,&i);
- Le paramètre de la fonction est l’adresse de la variable
- Passage par adresse = modification possible du paramètre

V4,0 - Copyright : Yantra Technologies 2004-2015

6.12 - Passage de paramètres par variable ou par valeur

Passage de paramètres par valeur

```
double mult( double x1, double y1 )
{
    y1 = x1 * y1;
    return y1;
}

void main()
{
    double x = 2;
     double y = 3;
    double z = mult(x,y);
   /*en sortie z = 6 , x=2 , y=3*/
}
```

Passage de paramètres par variable

```
double mult( const double* const  x1,
    double* y1 )
{
    *y1 = (*x1) * (*y1);
    return *y1;
}

void main()
{
    double x = 2;
     double y = 3;
    double z = mult(&x,&y);
   /*en sortie z = 6 , x=2 ,
     Attention y a été modifié y=6*/
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.13 - Pointeur de fonctions

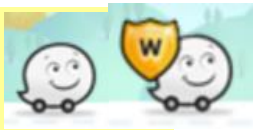
type (*identificateur) (paramètres);

Exemple :

```
double (*pf)(double, double);
pf =  mult;
double z = pf (2.,3.);

/* pf est un pointeur de fonction */.

typedef int (*PtrFonct)(int, int);
PtrFonct pf;
```


V4,0 - Copyright : Yantra Technologies 2004-2015

6.14 - Conclusion



Les pointeurs sont très utilisés en C, mais ils sont très dangereux, quelques règles de survie :

Règle 1 : TOUJOURS INITIALISER LES POINTEURS QUE VOUS UTILISEZ.

Règle 2 : VÉRIFIEZ QUE TOUTE DEMANDE D'ALLOCATION MÉMOIRE A ÉTÉ SATISFAITE.

Règle 3 : LORSQU'ON UTILISE UN POINTEUR, IL FAUT VÉRIFIER S'IL EST VALIDE.

V4,0 - Copyright : Yantra Technologies 2004-2015

7 - Les tableaux et les chaînes de caractères

7.1 - Les Tableaux
7.2 - Déclaration
7.3 - Conversions des tableaux en pointeurs
7.4 - Pointeurs, tableaux
7.5 - Tableaux à 1 dimension
7.6 - Tableaux à 2 dimensions
7.7 - Les chaînes de caractères
7.8 - Affichage et saisie d ’une chaîne

V4,0 - Copyright : Yantra Technologies 2004-2015

7.1 – Les Tableaux

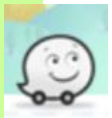


Tableau caractérisé par

- sa taille
- ses éléments repérés par son indice
- commence à 0 finit à dim-1

Tableau à une dimension

- Déclaration : Type nom[dim]; /* réservation de dim places de grandeur Type */
- Exemples
  - int Tab[10];
  - float vecteur[20];

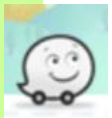
Tableau à deux dimensions

- Déclaration : type nom[dim1][dim2];
  - int compteur[4][5];
  - float nombre[2][10];

- Utilisation : nom[indice1][indice2];
  - compteur[3][4]=3;
  - printf("%d",compteur[1][2]);

V4,0 - Copyright : Yantra Technologies 2004-2015

7.2 - Déclaration



On peut initialiser au moment de la déclaration

- Exemple
  - **int liste[10]={1,2,45,5,67,120,32,48,3,10};**
  - **int tab[2][3]={{1,4,8},{8,5,3}}**

Remarque

- Consomment beaucoup de place
- Nécessité de les dimensionner au plus juste

V4,0 - Copyright : Yantra Technologies 2004-2015

Accès aux éléments d'un tableau par pointeur

```
int tableau[100];
int *pi=tableau;

tableau[3]=5;      /* Le 4ème élément est initialisé à 5 */
*(tableau+2)=4; /* Le 3ème élément est initialisé à 4 */
pi[5]=1;           /* Le 6ème élément est initialisé à 1 */
```


V4,0 - Copyright : Yantra Technologies 2004-2015

7.4 - Pointeurs, tableaux

Déclaration d'un pointeur (adresse du premier élément) + allocation de l 'espace mémoire
On peut donc déclarer nous-même un pointeur et allouer la mémoire ' manuellement '

```
int *tab = (int*) malloc(sizeof(int)*5)
```

Avantage : allocation de l 'espace mémoire au moment de l 'exécution, attention ne pas oublier de détruire la mémoire:

```
free(tab) ;
```

Il est possible de pointer sur un élément particulier d'un tableau :

```
int tab[5];
int *ptr = &tab[2];    /* ptr pointe sur le 3eme élément du tableau */
```

Un nom de tableau utilisé dans une expression est converti en une adresse du début de ce tableau :

```
int tab[5];
int *ptr = tab;      /* équivalent a : "ptr = &tab[0]" */
```


V4,0 - Copyright : Yantra Technologies 2004-2015

7.5 - Tableaux à 1 dimension

- Equivalences :

  » **int *tableau**

- int tableau[10];

  tableau=(int*)malloc(sizeof(int)*10);

  » ***tableau**

  tableau[0]

  » ***(tableau+i)**

  tableau[i]

  » **tableau**

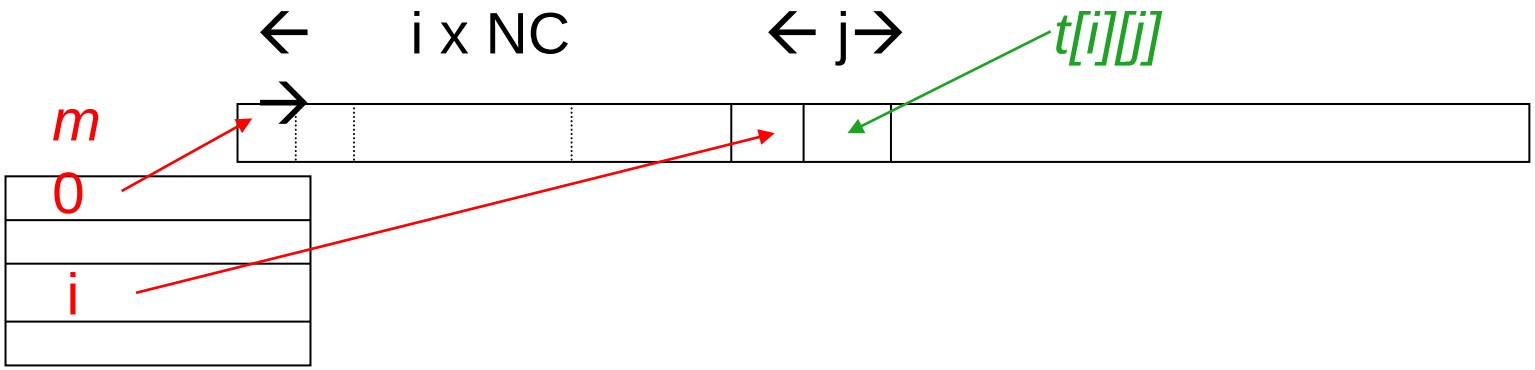
  &tableau[0]

  » **(tableau+i)**

  &tableau[i]

V4,0 - Copyright : Yantra Technologies 2004-2015

7.6 - Tableaux à 2 dimensions



- Equivalences :
  - » t[0] &t[0][0]
  - » t[1] &t[1][0]
  - » t[i] &t[i][0]
  - » t[i]+1 &(t[i][0])+1

t[NL][NC] , *m[NL]

t[i][j]-> t+i*NBCOL+j;

V4,0 - Copyright : Yantra Technologies 2004-2015

7.7 - Les chaînes de caractères

- Une chaîne = un tableau de caractères
- Déclaration
  - **char nom[dim]; ou char* nom;**
    nom=(char*)malloc(dim);
  - **exemple : char texte[10]; ou char* texte;**
    texte=(char*)malloc(10);
- Seulement dim-1 caractères disponibles
- Caractère NUL (‘ \0 ’) à la fin

V4,0 - Copyright : Yantra Technologies 2004-2015

7.8 - Affichage et saisie d ’une chaîne

- printf et format %s
  - Exemple : **char texte[10]=“bonjour“;**
    printf(“voici le texte %s“,texte);

- scanf et format %s
  - Exemple : **char texte[10];**
    scanf(“%s“,texte);

V4,0 - Copyright : Yantra Technologies 2004-2015

8 – Les fonctions

8.1 - Généralités
8.2 - Variables globales
8.3 - Variables locales
8.4 - Fonctions typées sans arguments
8.5 - Fonctions avec arguments
8.6 - Arguments par adresse
8.7 - Utilisation dans le programme
8.8 - Utilisation des entêtes .h
8.9 - Fonction main
8.10 - main : passage de paramètres

V4,0 - Copyright : Yantra Technologies 2004-2015

8.1 - Généralités

- En C procédures = fonctions
- Pas de déclaration de fonction dans une autre fonction
- Possible appel d ’une fonction dans une autre fonction
- Variables locales
- Variables globales

V4,0 - Copyright : Yantra Technologies 2004-2015

La déclaration d'une fonction

```
type_de_retour Nom_de_la_Fonction
( type1 argument1, type2 argument2, ... )
{
        liste d'instructions ;
}
```

```
int add( int i, int j) {
        int k = i + j ;
        return k;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

8.2 - Variables globales

- Déclarées avant le main
- Eviter le surnombre

- Exemple :

```
#include <stdio.h>
int i;
int main () { ...
```


V4,0 - Copyright : Yantra Technologies 2004-2015

8.3 - Variables locales

- Locales aux fonctions : connues uniquement à l ’intérieur de la fonction.
- Locales au main : connues dans le main mais pas à l ’intérieur des fonctions

V4,0 - Copyright : Yantra Technologies 2004-2015

8.4 - Fonctions typées sans arguments

- Fonction renvoyant une valeur
- Type de la valeur déclaré avec la fonction
- Valeur retournée spécifiée grâce à return

V4,0 - Copyright : Yantra Technologies 2004-2015

8.5 - Fonctions avec arguments

- Type void ou non
- Utilisation des valeurs de certaines variables du programme
- En-tête : variables arguments
  - Exemple : **int truc(int); /* prototype réduit */**
    int truc(int x); /* prototype complet */

V4,0 - Copyright : Yantra Technologies 2004-2015

8.6- Arguments par adresse

- Nécessaire pour modifier la variable
- Déclaration en-tête : utiliser les pointeurs
- Tableaux et Pointeurs

- Exemples :
  - Déclaration

    void Calcul(int x,int* y);
  - Appel
  - int j = 0; **int * k = &j;**

    Calcul(i,&j);

    Calcul(i,k);

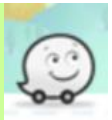
V4,0 - Copyright : Yantra Technologies 2004-2015

8.7 - Utilisation dans le programme

```
#include <stdio.h>
#include "produit.h"
 /*int produit (int, int); */

 int main () {
    int a, b,c;
    scanf("%d",&a); scanf("%d",&b); c = produit(a,b);
    printf("\nle produit vaut %d\n",c);
    return 0;
 }
/* produit.c
 int produit (int a, int b){return a*b;}*/
```


V4,0 - Copyright : Yantra Technologies 2004-2015

8.8 - Utilisation des entêtes .h



```
/***************************/
/***     fichier: produit.h      ***/
/***   produit de 2 entiers    ***/
/***************************/
#ifndef PRODUIT_H
#define PRODUIT_H
   int produit (int, int);

   int plus(int,int);
#endif  /* PRODUIT_H */
```


V4,0 - Copyright : Yantra Technologies 2004-2015

8.8 - Utilisation des entêtes .h

```
/*************************/
/***    fichier: produit.c     ***/
/***   produit de 2 entiers    ***/
/*************************/
#include  « produit.h »
int produit (int a, int b)
 {
    return a*b;
 }
```

```
/*************************/
/***    fichier: plus.c     ***/
/***   produit de 2 entiers    ***/
/*************************/
#include  « produit.h »
int plus(int a, int b)
 {
    return a+b;
 }
```


V4,0 - Copyright : Yantra Technologies 2004-2015

8.9 - Fonction main

```
int main() /* Plus petit programme C. */
{
    return 0;
}
```

La fonction main est une fonction spéciale, lorsqu'un programme est chargé, son exécution commence par l'appel de celle-ci, elle marque le début du programme.

Remarque :

Elle ne peut pas être récursive.

V4,0 - Copyright : Yantra Technologies 2004-2015

8.10 - main : passage de paramètres

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

int main (int argc, char* argv [], char* envp[] )
{
  int i=0;

  printf("Nb argument : %d \n",argc);

  for (i=0;i<argc;i++)
          printf("argv[%d]=%s \n",i,argv[i]);

  printf("\n variables d'environnement.\n");

  for (i=0;envp[i] != NULL ;i++)
          printf("envp[%d]= %s \n",i,envp[i]);
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

9.1 – Qualificatif const & volatile

Qualificatif **const**

Une variable dont le type est qualifiée par **const** ne peut pas être modifiée.

But : se protéger contre une erreur de programmation.

Ceci est utile pour les paramètres d'une fonction, lorsqu'on désire protéger les paramètres effectifs de la fonction en les mettant en **lecture seulement** pour la fonction.

V4,0 - Copyright : Yantra Technologies 2004-2015

9.1 – Qualificatif const & volatile

2 solutions :

- const type variable = valeur;
  - exemple : const float PI = 3.14;

- #define PI 3.14
  - utilisation du préprocesseur du compilateur. Le symbole PI sera remplacé par 3.14 dans tout le code.

const char c;	caractère constant
const char *p;	pointeur vers caractère constant
char * const p;	pointeur constant vers caractère
const char * const p;	pointeur constant vers caractère constant

V4,0 - Copyright : Yantra Technologies 2004-2015

9.2 – Qualificatif const & volatile

Qualificatif **volatile**

En qualifiant par ***volatile*** le type d'une variable, le compilateur par du principe que cette variable peut être modifiée par un moyen extérieur au programme.

Ceci se produit lorsqu'on interagit avec des parties matérielles de la machine : coupleurs d'entrées-sorties généralement.

Lorsqu'une variable est de type volatile le compilateur ne réalise pas d'optimisations qu'il réalise normalement sur les variables non volatiles.

Les qualificatifs de type deviennent pénibles en conjonction avec les pointeurs car on a les trois possibilités :

- l'objet pointé est qualifié
- le pointeur lui-même est qualifié ;
- le pointeur et l'objet pointé sont qualifiés.

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

int main()
{
    volatile int a = 1;
    volatile int b = 2;
    volatile int c = 3;
    volatile int d = 4;

    printf("c = %i\n",c);
    return 0;
}
```

https://c.developpez.com/cours/bernard-cassagne/node117.php

V4,0 - Copyright : Yantra Technologies 2004-2015

10 - Saisie de nombres et de caractères

10.1 - Les Chaines de Caractère

10.2 - Fonction printf

10.3 - Fonction scanf

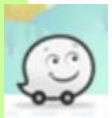
10.4 - Écriture d'un caractère : putchar

10.5 - Lecture d'un caractère

10.6 - Travailler avec des chaînes de caractères

V4,0 - Copyright : Yantra Technologies 2004-2015

10.1 – Les Chaines de Caractère



Une chaîne de caractères (appelée *string* en anglais) est une suite de caractères, défini par le code ASCII. (http://www.table-ascii.com/)

Remarque : Il n'existe pas de type spécial chaîne ou string en C

En langage C, une chaîne de caractères est un tableau, comportant plusieurs données de type **char**, dont le dernier élément est le caractère **nul '\0'**, c'est-à-dire le premier caractère du **code ASCII** (dont la valeur est 0).

Il existe des notations particulières et des fonctions spéciales pour le traitement de tableaux de caractères.

V4,0 - Copyright : Yantra Technologies 2004-2015

10.1 – Les Chaines de Caractère -> Déclaration

Déclaration de chaînes de caractères en C

```
char <NomVariable> [<Longueur>];
```

Exemples

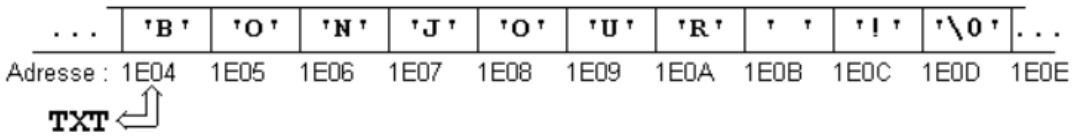
```
char NOM [20];
char PRENOM [20];
char PHRASE [300];
```

Le nom d'une chaîne est le représentant de ***l'adresse du premier caractère*** de la chaîne.

La représentation interne d'une chaîne de caractères est terminée par le symbole '\0' (NULL)

Donc pour mémoriser une variable qui doit être capable de contenir un texte de N caractères, nous avons besoin de N+1 octets en mémoire

```
char TXT[10] = "BONJOUR !";
```



https://www.ltam.lu/Tutoriel_Ansi_C/prg-c74.htm

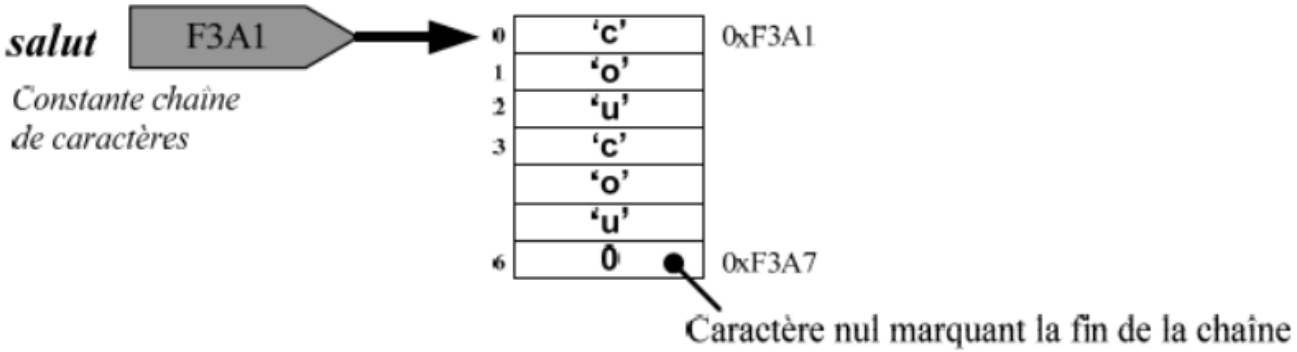
V4,0 - Copyright : Yantra Technologies 2004-2015

10.1 – Les Chaines de Caractère
-> Les chaînes de caractères constantes

const char* salut = **"**coucou**"** ;

const char salut[] = {'c','o', 'u','c','o', 'u', '****n'};

Le **compilateur** place la chaîne "coucou" dans une zone mémoire de 7 octets, qui contient les 6 lettres du mot coucou et le caractère nul 0.



V4,0 - Copyright : Yantra Technologies 2004-2015

10.1 – Les Chaines de Caractère -> Initialisation de chaînes de caractères

```
const char* salut = "coucou" ;
const char salut[] = {'c','o', 'u','c','o', 'u', '\n'};
char salut[] = {'c','o', 'u','c','o', 'u', '\n'};
char salut[7] = "coucou" ;
char salut[7] = {'c','o', 'u','c','o', 'u', '\n'};
char salut[10] = "coucou" ;
char salut[10] = {'c','o', 'u','c','o', 'u', '\n'};
```

char salut = "coucou" ; // Warning erreur possible à l'execution,*

char salut[5] = "coucou" ; // Erreur Compilation
char salut[5] = {'c','o', 'u','c','o', 'u', '\n'}; //Erreur Compilation

char salut[6] = "coucou" ; // Erreur Compilation ou a l'execution

V4,0 - Copyright : Yantra Technologies 2004-2015

10.1 – Les Chaines de Caractère -> Accès aux éléments d'une chaîne

Une chaine de caractères est un tableaux de caractère donc utilisation du [] ou du décalage de pointeur

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

int main ()
{
  char val[] = "Bonjour";

  for ( unsigned i(0) ; i < strlen(val); ++i)
  {
      printf("%c \n",val[i]);
  }
   printf("\n");
   for ( unsigned i(0) ; i < strlen(val); ++i)
  {
      ++val[i];
      printf("%c \n",val[i]);
  }
  return 0;
}
```

```
B
o
n
j
o
u
r

C
p
o
k
p
v
s
```

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

int main ()
{
  char val[] = "Bonjour";

  for ( unsigned i(0) ; i < strlen(val); ++i)
  {
      printf(" %c \n",*(val+i));
  }
   printf("\n");
   for ( unsigned i(0) ; i < strlen(val); ++i)
  {
      ++val[i];
      printf(" %c \n",*(val+i));
  }
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.1 – Les Chaines de Caractère -> Précédence alphabétique et lexicographique

La précédence des caractères dans l'alphabet d'une machine est dépendante du code de caractères utilisé.

Pour le code ASCII, nous pouvons constater l'ordre suivant:

. . . ,0,1,2, ... ,9, . . . ,A,B,C, ... ,Z, . . . ,a,b,c, ... ,z, . . .

V4,0 - Copyright : Yantra Technologies 2004-2015

10.2 – Fonction printf

Fonction de la bibliothèque <stdio.h>

printf de faire afficher une chaîne de caractères à l'écran.

int printf(const char* format, ...);

fprintf permet d'écrire dans les fichiers de la même manière que printf écrit à l'écran.

int fprintf(FILE* fd, const char* format, ...);

sprintf permet d'écrire dans un buffer de façon formatée.

int sprintf(char* buf, const char* format, ...);

vprintf prend un tableau d'arguments plutôt que des arguments à la suite les uns derrière les autres. D'autres variantes utilisent ce principe pour les services de sprintf ou fprintf : les fonctions sont alors vfprintf et vsprintf.

int vprintf(const char * format, va_list arg);

V4,0 - Copyright : Yantra Technologies 2004-2015

10.2 – Fonction printf

Code format	Type de l'argument à afficher	Format d'affichage et exemples
%c	caractère	Ex : a G ? +
%hd %hu %hX	entier *short int* (ou caractère) **signé** entier *short int* (ou caractère) **non signé** pour afficher en hexadécimal (*short* ou *char*)	Base 10. Ex : -12 Base 10. Ex : 463 Hexadécimal. Ex : 9A0F
%ld %lu %lX	entier *long int* **signé** entier *long int* **non signé** entier *long int* en hexadécimal	Base 10. Ex : -1289 Base 10. Ex : 46399 Hexadécimal. Ex : B4E98A0F
%d %X	entier *int* pour afficher un *int* en hexadécimal	Base 10. Ex : -546 Hexadécimal. Ex : 9A0F
%lf %le	Réel double précision (*double*)	Virgule flottante. Ex : -3.141592 Avec exposant. Ex : -1.450000e-7
%f %e	Réel simple précision (*float*)	Virgule flottante. Ex : -3.141592 Avec exposant. Ex : -1.450000e-7
%s	Chaîne de caractères	Ex : bonjour !

Code format	Code ASCII	Caractère	Effet à l'écran
\n \t \r	10 9 13	*LF* *HT* *CR*	Changement de ligne Avance d'une tabulation horizontale Retour en début de ligne courante
%% \\ \"		% \ "	Écrit le caractère à l'écran. Ces caractères ne sont pas utilisés comme les autres, car ils ont une signification particulière dans *printf*.
\x*HHH*	0xHHH	n'importe lequel	HHH = valeur hexa du code ASCII

https://fr.wikibooks.org/wiki/Les_ASCII_de_0_%C3%A0_127/La_table_ASCII

V4,0 - Copyright : Yantra Technologies 2004-2015

10.2 – Fonction printf

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

int main()
{
   printf ("Characters: %c %c \n", 'a', 65);
   printf ("Decimals: %d %ld\n", 1977, 650000L);
   printf ("Preceding with blanks: %10d \n", 1977);
   printf ("Preceding with zeros: %010d \n", 1977);
   printf ("Some different radices: %d %x %o %#x %#o \n", 100, 100, 100, 100, 100);
   printf ("floats: %4.2f %+.0e %E \n", 3.1416, 3.1416, 3.1416);
   printf ("Width trick: %*d \n", 5, 10);
   printf ("%s \n", "A string");
   return 0;
}
```

```
Characters: a A
Decimals: 1977 650000
Preceding with blanks:       1977
Preceding with zeros: 0000001977
Some different radices: 100 64 144 0x64 0144
floats: 3.14 +3e+00 3.141600E+00
Width trick:    10
A string
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf

Fonction de la bibliothèque <stdio.h>

fscanf permet de lire dans les fichiers de la même manière que scanf lit au clavier.

int fscanf(FILE* fd, const char* format, …);

sscanf permet de lire dans un buffer (tampon) de façon formatée.

int sscanf(char* buf, const char* format, …);

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf

SYMBOLE	LECTURE D'UN(E)	TYPE
%d ou %i	entier relatif	int*
%u	entier naturel (unsigned)	int*
%o	entier exprimé en octal	int*
%b	entier exprimé en hexadécimal	int*
%c	caractère	char*
%s	chaîne de caractères	char*
%f ou %e	rationnel en notation décimale ou exponentielle (scientifique)	float*

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf

```c
#include <stdio.h>

int main ()
{
  char str [80];
  int i;

  printf ("Enter your family name: ");
  scanf ("%79s",str);
  printf ("Enter your age: ");
  scanf ("%d",&i);
  printf ("Mr. %s , %d years old.\n",str,i);
  printf ("Enter a hexadecimal number: ");
  scanf ("%x",&i);
  printf ("You have entered %#x (%d).\n",i,i);

  return 0;
}
```

Les variables modifiées sont précédées de **&** sauf les tableaux et pointeurs

```
Enter your family name: david
Enter your age: 48
Mr. david , 48 years old.
Enter a hexadecimal number: 0x125
You have entered 0x125 (293).
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf

```
char car;
int i;
float r;
char chaine[100];

scanf("%c",&car);  /* saisie d'un caractère */
scanf("%d",&i);     /* saisie d'un entier */
scanf("%f",&r);      /* saisie d'un nombre réel */
scanf("%10s",chaine);      /* saisie d'une chaine de caractère  */
```

Remarque:
format non respecté, saisie ignorée

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf

%[*][width][modifiers]type

http://www.cplusplus.com/reference/clibrary/cstdio/scanf/

where:

*	An optional starting asterisk indicates that the data is to be retrieved from `stdin` but ignored, i.e. it is not stored in the corresponding argument.
width	Specifies the maximum number of characters to be read in the current reading operation
modifiers	Specifies a size different from `int` (in the case of `d`, `i` and `n`), `unsigned int` (in the case of `o`, `u` and `x`) or `float` (in the case of `e`, `f` and `g`) for the data pointed by the corresponding additional argument: **h :** `short int` (for `d`, `i` and `n`), or `unsigned short int` (for `o`, `u` and `x`) **l :** `long int` (for `d`, `i` and `n`), or `unsigned long int` (for `o`, `u` and `x`), or `double` (for `e`, `f` and `g`) **L :** `long double` (for `e`, `f` and `g`)
type	A character specifying the type of data to be read and how it is expected to be read. See next table.

scanf type specifiers:

type	Qualifying Input	Type of argument
c	**Single character:** Reads the next character. If a *width* different from 1 is specified, the function reads *width* characters and stores them in the successive locations of the array passed as argument. No null character is appended at the end.	`char *`
d	**Decimal integer:** Number optionally preceded with a + or – sign.	`int *`
e,E,f,g,G	**Floating point:** Decimal number containing a decimal point, optionally preceded by a + or – sign and optionally folowed by the `e` or `E` character and a decimal number. Two examples of valid entries are -732.103 and 7.12e4	`float *`
o	**Octal integer.**	`int *`
s	**String of characters.** This will read subsequent characters until a whitespace is found (whitespace characters are considered to be blank, newline and tab).	`char *`
u	Unsigned decimal integer.	`unsigned int *`
x,X	Hexadecimal integer.	`int *`

V4,0 - Copyright : Yantra Technologies 2004-2015

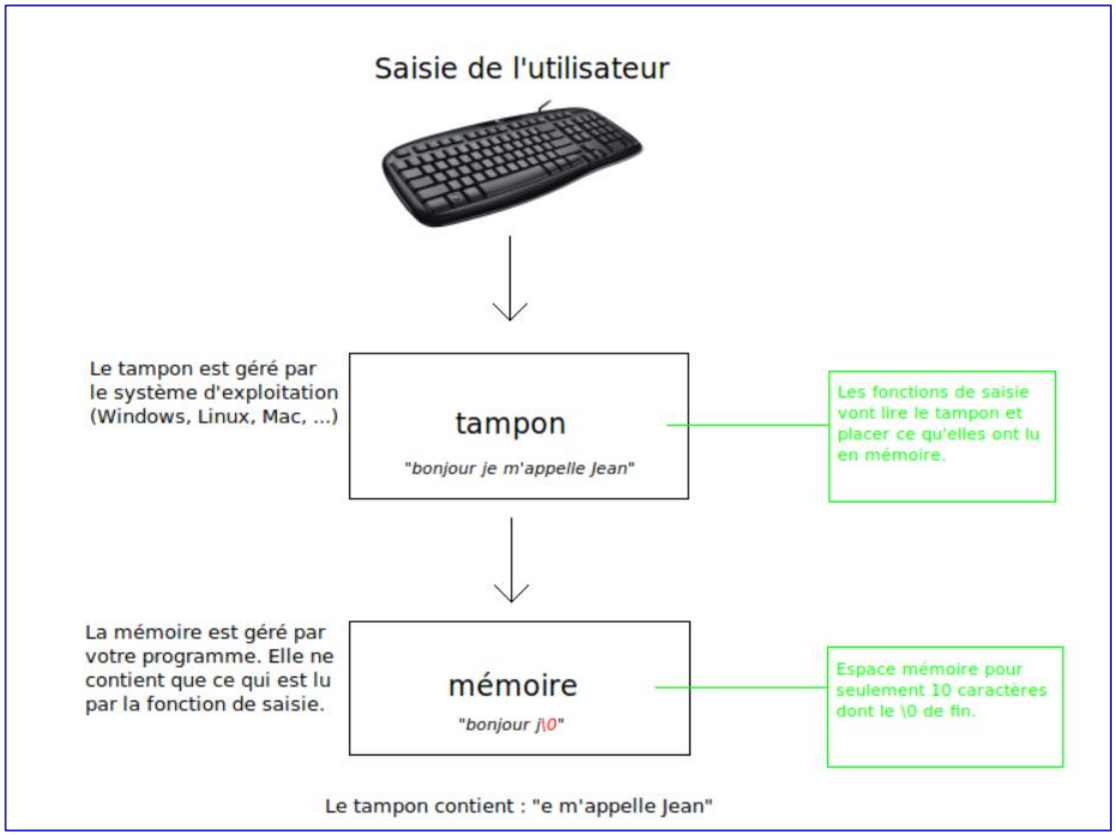
10.3 – Fonction scanf -> Sécuriser scanf

- Saisie clavier -> stockage dans buffer
- Buffer de type FIFO (First In First Out)
- Dernier caractère stocké : LF (0x0A) (\n)
- Buffer appelé Flux D'entrée
- Compilation -> vide le flux d'entrée

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf -> Sécuriser scanf

```
char chaine[6];
printf ("Tapez 5 caractères : ");
scanf ("%5s", chaine);
/* que ce passe-t-il si l 'utilisateur tape 10 caractère ? */
```

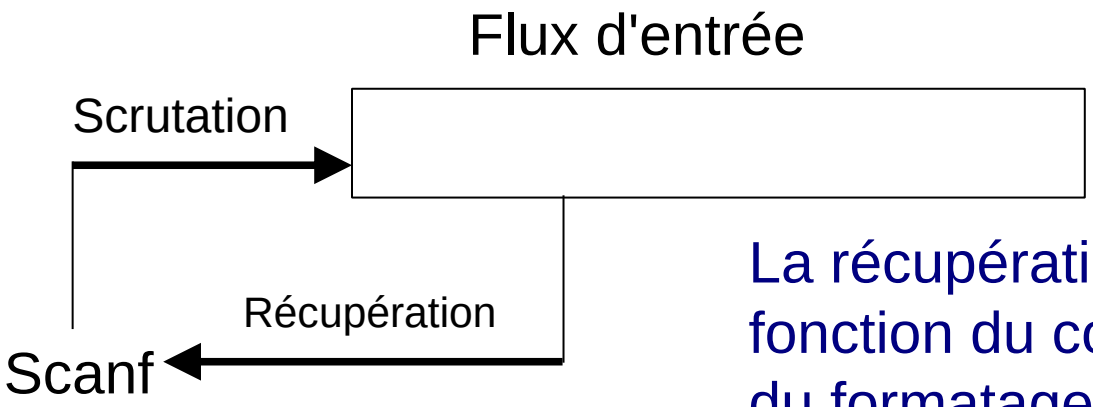


https://pub.phyks.me/sdz/sdz/la-saisie-securisee-avec-scanf.html

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf -> Sécuriser scanf

- À l'appel, la fonction scanf scrute le flux



La récupération se fait en fonction du contenu du buffer et du formatage du scanf

- Problèmes liés à LF !!
  - Exemples : saisie de 2 caractères à la suite, caractère puis entier, etc.

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf -> Sécuriser scanf

Problématique :

L'utilisateur tape ceci :
" 123azerty456uiop789 "

On ne veut récupérer uniquement les premiers et derniers chiffres (123 et 789)

La syntaxe est donc la suivante :
[nombre]~[lettres]~[nombre]~[lettres]~[nombre]

```
int a, b;
scanf ("%d%*[a-z]%*[0-9]%*[a-z]%d", &a, &b);
```

https://nicolasj.developpez.com/articles/regex/#LIII-D
https://pub.phyks.me/sdz/sdz/la-saisie-securisee-avec-scanf.html

V4,0 - Copyright : Yantra Technologies 2004-2015

10.3 – Fonction scanf -> Sécuriser scanf

%n retourne le nombre de caractères lus par scanf() dans une variable de type **int**, celui ci n'est pas compté non plus dans la valeur de retour de scanf().
Elle permet donc une meilleure précision sur la gestion d'erreur :

```
int a, b;
int seq0, seq1, seq2, seq3, seq4;
int ret;

ret = scanf ("%d%n%*[a-z]%n%*[0-9]%n%*[a-z]%n%d%n", &a, &seq0, &seq1, &seq2, &seq3, &b, &seq4);

/* vidage du buffer */
scanf ("%*[^\n");
getchar ();

/* vérification */
if (ret != 2)
{
   if (!seq0) /* premier nombre qui a échoué */
   if (!seq1) /* première séquence de lettre qui a échoué */
   if (!seq2) /* la séquence de chiffre a échoué */
   if (!seq3) /* la dernière séquence de lettre a échoué */
   if (!seq4) /* dernier nombre qui à échoué */
}
```

https://pub.phyks.me/sdz/sdz/la-saisie-securisee-avec-scanf.html

V4,0 - Copyright : Yantra Technologies 2004-2015

10.4 - Écriture d'un caractère : putchar()

putchar() est une fonction qui écrit le caractère passé en argument sur la sortie standard (l'écran).

int putchar (int character);

```
#include <stdio.h>

int main ()
{
  char c;
  for (c = 'A' ; c <= 'Z' ; c++) putchar (c);

  return 0;
}
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.5 - Lecture d'un caractère : getchar()

getchar() est une fonction de <stdio.h> sans paramètre qui « capte » la valeur du caractère lu à la position courante de l'entrée standard (le clavier) et le retourne. Moins gourmande que scanf

int getchar (void);

```
#include <stdio.h>

int main ()
{
  int c;
  puts ("Enter text. Include a dot ('.') in a sentence to exit:");
  do {
    c=getchar();
    putchar (c);
  } while (c != '.');
  return 0;
}
```

```
Enter text. Include a dot ('.') in a sentence to exit:
bonjour .
bonjour .
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.5 - Lecture d'un caractère : fgetc

fgetc() est une fonction permettant toutes les trois de lire un caractère à la position courante du flux de caractères considéré

```
int fgetc( FILE * stream );
```

⇔

```
int getc( FILE * stream );
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.5 - Lecture d'un caractère : fgetc

Les 2 fonctions fgets() permettent de lire le contenu d'une chaîne de caractères à partir d'un flux,

la fonction gets() a le flux de lecture imposé : stdin.

```
char * fgets( char * string, int maxLength, FILE * stream ) // Avant C99
char * fgets( char * restrict string, int maxLength, FILE * restrict stream ); // A partir de C99

char * gets( char * string ); // Supprimé à partir de C11
```


V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères

- Les fonctions de <stdio.h>
- Les fonctions de <string>
- Les fonctions de <stdlib>
- Les fonctions de <ctype>

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <stdio.h>

« **Standard Input/Output Header** » ou « En-tête Standard d'Entrée/Sortie », est l'en-tête de la bibliothèque standard du C déclarant les macros, les constantes et les définitions de fonctions utilisées dans les opérations d'entrée/sortie.

Elle permet principalement la manipulation des flux de caractères (que ce soit vers un fichier ou la console). Cette librairie propose aussi un ensemble de fonctions utiles pour la gestion de vos fichiers.

V4,0 - Copyright : Yantra Technologies 2004-2015

Opérations sur les fichiers

remove	Remove file (function)
rename	Rename file (function)
tmpfile	Open a temporary file (function)
tmpnam	Generate temporary filename (function)

Opérations d'accès aux fichiers

fclose	Close file (function)
fflush	Flush stream (function)
fopen	Open file (function)
freopen	Reopen stream with different file or mode (function)
setbuf	Set stream buffer (function)
setvbuf	Change stream buffering (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

Le formatage des entrées sorties

fprintf	Write formatted data to stream (function)
fscanf	Read formatted data from stream (function)
printf	Print formatted data to stdout (function)
scanf	Read formatted data from stdin (function)
snprintf C++11	Write formatted output to sized buffer (function)
sprintf	Write formatted data to string (function)
sscanf	Read formatted data from string (function)
vfprintf	Write formatted data from variable argument list to stream (function)
vfscanf C++11	Read formatted data from stream into variable argument list (function)
vprintf	Print formatted data from variable argument list to stdout (function)
vscanf C++11	Read formatted data into variable argument list (function)
vsnprintf C++11	Write formatted data from variable argument list to sized buffer (function)
vsprintf	Write formatted data from variable argument list to string (function)
vsscanf C++11	Read formatted data from string into variable argument list (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

Gestion des caractères en entrées / sorties

fgetc	Get character from stream (function)
fgets	Get string from stream (function)
fputc	Write character to stream (function)
fputs	Write string to stream (function)
getc	Get character from stream (function)
getchar	Get character from stdin (function)
gets	Get string from stdin (function)
putc	Write character to stream (function)
putchar	Write character to stdout (function)
puts	Write string to stdout (function)
ungetc	Unget character from stream (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <string>

La bibliothèque <*string*> fournit une multitude de fonctions pratiques pour le traitement de chaînes de caractères.

V4,0 - Copyright : Yantra Technologies 2004-2015

Fonctions de copie

memcpy	Copy block of memory (function)
memmove	Move block of memory (function)
strcpy	Copy string (function)
strncpy	Copy characters from string (function)

Fonction de concaténation

strcat	Concatenate strings (function)
strncat	Append characters from string (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

Fonction de comparaison

memcmp	Compare two blocks of memory (function)
strcmp	Compare two strings (function)
strcoll	Compare two strings using locale (function)
strncmp	Compare characters of two strings (function)
strxfrm	Transform string using locale (function)

Fonction de recherche

memchr	Locate character in block of memory (function)
strchr	Locate first occurrence of character in string (function)
strcspn	Get span until character in string (function)
strpbrk	Locate characters in string (function)
strrchr	Locate last occurrence of character in string (function)
strspn	Get span of character set in string (function)
strstr	Locate substring (function)
strtok	Split string into tokens (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <stdlib.h>

La librairie <stdlib.h> (pour standard library) propose des fonctions pour répondre à différentes problématiques.

Elle permet de

- gérer la mémoire dynamiquement, en allouant ou libérant de la mémoire dans le tas (le heap en anglais).
- pouvoir interagir avec le système d'exploitation sur lequel le programme s'exécute.

Cette librairie contient aussi d'autres fonctions utilitaires pour les convertions entre types de données, des algorithmes (tri, recherche dichotomique, génération de nombres pseudo-aléatoires, ...) etc…

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <stdlib.h>

Fonction de conversion

atof	Convert string to double (function)
atoi	Convert string to integer (function)
atol	Convert string to long integer (function)
atoll C++11	Convert string to long long integer (function)
strtod	Convert string to double (function)
strtof C++11	Convert string to float (function)
strtol	Convert string to long integer (function)
strtold C++11	Convert string to long double (function)
strtoll C++11	Convert string to long long integer (function)
strtoul	Convert string to unsigned long integer (function)
strtoull C++11	Convert string to unsigned long long integer (function)

Génération d'un nombre aléatoire

rand	Generate random number (function)
srand	Initialize random number generator (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

Gestion de la mémoire

calloc	Allocate and zero-initialize array (function)
free	Deallocate memory block (function)
malloc	Allocate memory block (function)
realloc	Reallocate memory block (function)

Environnement

abort	Abort current process (function)
atexit	Set function to be executed on exit (function)
at_quick_exit C++11	Set function to be executed on quick exit (function)
exit	Terminate calling process (function)
getenv	Get environment string (function)
quick_exit C++11	Terminate calling process quick (function)
system	Execute system command (function)
_Exit C++11	Terminate calling process (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <stdlib.h>

Fonctions de recherche

bsearch	Binary search in array (function)
qsort	Sort elements of array (function)

Fonctions arithmétiques

abs	Absolute value (function)
div	Integral division (function)
labs	Absolute value (function)
ldiv	Integral division (function)
llabs C++11	Absolute value (function)
lldiv C++11	Integral division (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

Fonctions liées aux caractères étendues

mblen	Get length of multibyte character (function)
mbtowc	Convert multibyte sequence to wide character (function)
wctomb	Convert wide character to multibyte sequence (function)

Fonction liées aux chaines de caractères étendus

mbstowcs	Convert multibyte string to wide-character string (function
wcstombs	Convert wide-character string to multibyte string (function

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <stdlib.h>

Macro

EXIT_FAILURE	Failure termination code (macro)
EXIT_SUCCESS	Success termination code (macro)
MB_CUR_MAX	Maximum size of multibyte characters (macro)
NULL	Null pointer (macro)
RAND_MAX	Maximum value returned by rand (macro)

Types

div_t	Structure returned by div (type)
ldiv_t	Structure returned by ldiv (type)
lldiv_t C++11	Structure returned by lldiv (type)
size_t	Unsigned integral type (type)

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <ctype.h>

Les fonctions de <*ctype.h*> servent à classifier et à convertir des caractères.

Les symboles nationaux (é, è, ä, ü, ß, ç, ...) ne sont pas considérés.

Les fonctions de <*ctype.h*> sont indépendantes du code de caractères de la machine et favorisent la portabilité des programmes.

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <ctype.h>

ASCII values	characters	iscntrl	isblank	isspace	isupper	islower	isalpha	isdigit	isxdigit	isalnum	ispunct	isgraph	isprint
0x00 .. 0x08	NUL, (other control codes)	x											
0x09	tab ('\t')	x	x	x									
0x0A .. 0x0D	(white-space control codes: '\f',' \v','\n','\r')	x		x									
0x0E .. 0x1F	(other control codes)	x											
0x20	space (' ')		x	x									x
0x21 .. 0x2F	!"#$%&'()*+,-./										x	x	x
0x30 .. 0x39	0123456789							x	x	x		x	x
0x3a .. 0x40	:;<=>?@										x	x	x
0x41 .. 0x46	ABCDEF				x		x		x	x		x	x
0x47 .. 0x5A	GHIJKLMNOPQRSTUVWXYZ				x		x			x		x	x
0x5B .. 0x60	[\]^_`										x	x	x
0x61 .. 0x66	abcdef					x	x		x	x		x	x
0x67 .. 0x7A	ghijklmnopqrstuvwxyz					x	x			x		x	x
0x7B .. 0x7E	{|}~										x	x	x
0x7F	(DEL)	x											

V4,0 - Copyright : Yantra Technologies 2004-2015

10.6 - Travailler avec des chaînes de caractères : <ctype.h>

Fonctions de test

isalnum	Check if character is alphanumeric (function)
isalpha	Check if character is alphabetic (function)
isblank C++11	Check if character is blank (function)
iscntrl	Check if character is a control character (function)
isdigit	Check if character is decimal digit (function)
isgraph	Check if character has graphical representation (function)
islower	Check if character is lowercase letter (function)
isprint	Check if character is printable (function)
ispunct	Check if character is a punctuation character (function)
isspace	Check if character is a white-space (function)
isupper	Check if character is uppercase letter (function)
isxdigit	Check if character is hexadecimal digit (function)

Fonctions de conversion

tolower	Convert uppercase letter to lowercase (function)
toupper	Convert lowercase letter to uppercase (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

11 - Types de variables complexes

11.1 - Types synonymes : typedef

11.2 - Structures

11.3 - Accéder aux champs

11.4 - Structures et pointeurs

11.5 – Le type union

11.6 - Exemple : Type union

11.7 – Le type enum

V4,0 - Copyright : Yantra Technologies 2004-2015

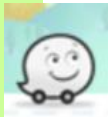
11.1 - Types synonymes : typedef

- Création de ses propres types
- Déclaration en début de programme

- Exemple
  - **typedef int entier; /* type entier = int */**
  - **typedef int Type_tab[5]; /* type Type_tab = tableaux de 5 entiers */**
  - **typedef char* fcar; /* type fcar = pointeur de char */**
  - **typedef int* vecteurInt;**
  - **typedef vecteurInt* matriceInt;**

V4,0 - Copyright : Yantra Technologies 2004-2015

11.2 - Structures



- Types particuliers
- Composition d'un ensemble de types

typedef struct {
types; } nomdutype;

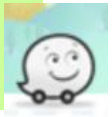
```
struct Nom_Structure {
        type_champ1 Nom_Champ1;

        type_champ2 Nom_Champ2;

        type_champ3 Nom_Champ3;

        type_champ4 Nom_Champ4;

        type_champ5 Nom_Champ5;

        ...
};
```

- Exemple

```
typedef struct personne
        {
                char* nom;
                char* prenom;
        } Personne;
Personne nom;        ⇔ struct personne nom;
```


V4,0 - Copyright : Yantra Technologies 2004-2015

11.3 - Accéder aux champs



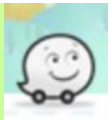
- Syntaxe : objet.champ
  - Exemple :

```
Personne toto;
toto.nom = (char*) malloc(sizeof(char)*10);
scanf("%9s",toto.nom);
free(toto.nom);
```

- Pas de protection sur les champs

V4,0 - Copyright : Yantra Technologies 2004-2015

11.4 - Structures et pointeurs



- Pointeurs sur des structures
  - allocation dynamique de structures
  - modification d'une structure dans une fonction
  - Exemple :

```
typedef struct { char* nom; char* prenom; unsigned short int age} Personne;
Personne* toto;
toto = (Personne*)malloc( sizeof(Personne));
(*toto).age=12;
free(toto);
```

- Notation équivalente : ->
  - Exemple : **(*toto).age** **toto->age**

V4,0 - Copyright : Yantra Technologies 2004-2015

11.5 – Le type union

Syntaxe :

```
union <nom de l'union>
    {
    <type de donnée> <nom de la variable>;
    ...
    <type de donnée> <nom de la variable>;
    };
```

Description:

Le type union permet de définir des variables qui occupent le **même emplacement mémoire.**

La taille d'une union est celle de la donnée la plus grande de cette union.

L'écriture dans une variable **écrase la valeur** des autres variables de l'union.

Il faut utiliser le sélecteur d'enregistrement (.) pour accéder aux éléments d'une union.

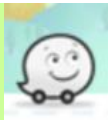
V4,0 - Copyright : Yantra Technologies 2004-2015

11.6 - Exemple : Type union

```
union Nombre
    {
    int i;
    double d;
    char cNombre[12];
    };

Nombre nb;
nb.d = 2.3
```


V4,0 - Copyright : Yantra Technologies 2004-2015

11.7 – Le type enum



Syntaxe :

enum [<nom de l'énumération>] {<nom d'une constante> [= <valeur>], ...} [<liste de variables>];

Description:

Le mot clé enum permet de définir un ensemble de constantes de type entier(int).
Une énumération fournit des identificateurs mnémoniques pour un ensemble de valeurs entières et finies
Une variable énumérée ne peut se voir affecter qu'un de ses énumérateurs(<constante>).
En l'absence d'une <valeur>, la première constante prend la valeur zéro.
Toute constante sans <valeur> sera augmentée d'un par rapport à la constante précédente.

Exemple :

```
enum jour { Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche};
enum jour { Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche} variable_jour;
enum { Lundi=1, Mardi=2, Mercredi=3, Jeudi=4, Vendredi=5, Samedi=6, Dimanche=7};
```


V4,0 - Copyright : Yantra Technologies 2004-2015

12 – Les Directives de Précompilation

Toutes les commandes du préprocesseur commencent :

- en début de ligne,
- par un signe dièse (#).

#include : inclusion de fichier

#include "fichier" ou #include <fichier>

#define POSIX_VERSION 1001 (en C++ const int POSIX_VERSION = 1001;)

V4,0 - Copyright : Yantra Technologies 2004-2015

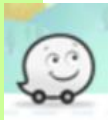
12 – Les Directives de Précompilation

Le préprocesseur définit un certain nombre de constantes :

Nom des macros	Développement de la macro associée
`__LINE__`	Se développe en une valeur numérique associée au numéro de la ligne courante dans le code du programme.
`__FILE__`	Cette macro sera remplacée par le nom du fichier en cours de traitement.
`__DATE__`	Celle-ci se transformera en une chaîne de caractères contenant la date de traitement du fichier sous un format "Mmm jj aaaa".
`__TIME__`	De même, se transformera en un chaîne représentant l'heure de traitement du fichier sous le format "hh:mm:ss".
`__STDC__`	Cette macro n'est définie que si votre compilateur respecte les normes ANSI ou ISO. Si tel est le cas, elle est alors définie à la valeur 1.
`__STDC_VERSION__`	Cette macro existe que depuis C99 (C ISO 19999) : elle n'est donc non définie pour les anciens compilateur. En C99 elle renverra la valeur `199901L` (typée long), en C11 elle renverra la valeur `201112L` et en C18 elle renverra la valeur `201710L` (oui, je sais, c'est bizarre).

V4,0 - Copyright : Yantra Technologies 2004-2015

12 – Les Directives de Précompilation



Compilation conditionnelle :

#ifdef identificateur

...

#endif

```
#ifndef _STDIO_H
#define _STDIO_H   1

...

#endif /* !_STDIO_H */
```

Protection contre l'inclusion multiple

d'autres directives de compilation conditionnelle :

- #ifndef (if not defined ...)
- #elif (sinon, si ...)
- #if (si ...)

autres commandes :

- # : ne fait rien (directive nulle) ;
- #error message : permet de stopper la compilation en affichant le message d'erreur donné en paramètre ;
- #line numéro [fichier] : permet de changer le numéro de ligne courant et le nom du fichier courant lors de la compilation ;
- #pragma texte : permet de donner des ordres spécifiques à une implémentation du compilateur

V4,0 - Copyright : Yantra Technologies 2004-2015

12 - Les Directives de Précompilation : Macro

#define nom valeur

#define macro(paramètre[, paramètre [...]]) définition

Exemple :

#define My_PI 3.14

#define MAX(x,y) ((x)>(y)?(x):(y))

#define MIN(x,y) ((x)<(y)?(x):(y))

#define CHAINE(s) #s // transforme son argument en chaîne de caractères.

#define NOMBRE(chiffre1,chiffre2) chiffre1##chiffre2 // permet de construire un nombre à deux chiffres.

V4,0 - Copyright : Yantra Technologies 2004-2015

12 - Les Directives de Précompilation : Exemple

```
#ifdef MON_DEBUG
#include <stdio.h>
#define TRACE(mes) { \
printf("Trace : %s [%d] : %s\n",__FILE__,__LINE__,mes); \
}
#else
#define TRACE(mes) {}
#endif
int main() {
TRACE("Message");
/*
{ printf("Trace : %s [%d] : %s\n",__FILE__,__LINE__,"Message); }
*/
return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13 - Les Fichiers

13.1 – Généralités
13.2 - Manipulation de fichiers
13.3 - Déclaration d'un fichier
13.3 - Déclaration d'un fichier
13.4 - Ouverture
13.5 - Fermeture
13.6 - Destruction
13.7 - Renommer
13.8 - Fichiers Binaires
13.9 - Fichiers Textes
13.10 - Gestion des erreurs

Réf : https://koor.fr/C/Index.wp **https://www.cplusplus.com/reference/**

V4,0 - Copyright : Yantra Technologies 2004-2015

13.1 - Les Fichiers : Généralités

- Fichier = ensemble d'informations stockées sur une mémoire de masse
- En C un fichier = une suite d'octets
- Les informations sont localisées par un pointeur
- 2 types d'accès : séquentiel et direct
- 2 types de codage : ASCII et binaires

V4,0 - Copyright : Yantra Technologies 2004-2015

13.2 - Les Fichiers : Manipulation de fichiers

La bibliothèque standard <stdio.h>. vous fournit différentes fonctions pour manipuler les fichiers.

Mais celles-ci manipulent non pas des fichiers, mais des flux de données en provenance ou à destination de fichiers. Ces flux peuvent être de deux types :

- des flux de textes qui sont des suites de caractères terminées par un caractère de fin de ligne (\n) et formant ainsi des lignes ;
- des flux binaires qui sont des suites de multiplets.

V4,0 - Copyright : Yantra Technologies 2004-2015

13.2 - Les Fichiers : Manipulation de fichiers

Fonction sur les fichiers

remove	Remove file (function)
rename	Rename file (function)
tmpfile	Open a temporary file (function)
tmpnam	Generate temporary filename (function)

Accéder aux fichiers

fclose	Close file (function)
fflush	Flush stream (function)
fopen	Open file (function)
freopen	Reopen stream with different file or mode (function)
setbuf	Set stream buffer (function)
setvbuf	Change stream buffering (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

Formatage input/output

fprintf	Write formatted data to stream (function)
fscanf	Read formatted data from stream (function)
printf	Print formatted data to stdout (function)
scanf	Read formatted data from stdin (function)
snprintf C++11	Write formatted output to sized buffer (function)
sprintf	Write formatted data to string (function)
sscanf	Read formatted data from string (function)
vfprintf	Write formatted data from variable argument list to stream (function)
vfscanf C++11	Read formatted data from stream into variable argument list (function)
vprintf	Print formatted data from variable argument list to stdout (function)
vscanf C++11	Read formatted data into variable argument list (function)
vsnprintf C++11	Write formatted data from variable argument list to sized buffer (function)
vsprintf	Write formatted data from variable argument list to string (function)
vsscanf C++11	Read formatted data from string into variable argument list (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

13.2 - Les Fichiers : Manipulation de fichiers

Caractère input/output

fgetc	Get character from stream (function)
fgets	Get string from stream (function)
fputc	Write character to stream (function)
fputs	Write string to stream (function)
getc	Get character from stream (function)
getchar	Get character from stdin (function)
gets	Get string from stdin (function)
putc	Write character to stream (function)
putchar	Write character to stdout (function)
puts	Write string to stdout (function)
ungetc	Unget character from stream (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

13.2 - Les Fichiers : Manipulation de fichiers

Fonction accès direct input/output

fread	Read block of data from stream (function)
fwrite	Write block of data to stream (function)

Positionnement dans les fichiers

fgetpos	Get current position in stream (function)
fseek	Reposition stream position indicator (function)
fsetpos	Set position indicator of stream (function)
ftell	Get current position in stream (function)
rewind	Set position of stream to the beginning (function)

Erreur

clearerr	Clear error indicators (function)
feof	Check end-of-file indicator (function)
ferror	Check error indicator (function)
perror	Print error message (function)

V4,0 - Copyright : Yantra Technologies 2004-2015

13.2 - Les Fichiers : Manipulation de fichiers

Macro

BUFSIZ	Buffer size (constant)
EOF	End-of-File (constant)
FILENAME_MAX	Maximum length of file names (constant)
FOPEN_MAX	Potential limit of simultaneous open streams (constant)
L_tmpnam	Minimum length for temporary file name (constant)
NULL	Null pointer (macro)
TMP_MAX	Number of temporary files (constant)

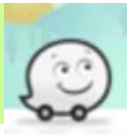
Additionally: _IOFBF, _IOLBF, _IONBF (used with setvbuf)
and SEEK_CUR, SEEK_END and SEEK_SET (used with fseek).

Types

FILE	Object containing information to control a stream (type)
fpos_t	Object containing information to specify a position within a file (type)
size_t	Unsigned integral type (type)

V4,0 - Copyright : Yantra Technologies 2004-2015

13.3 - Les Fichiers : Déclaration d'un fichier

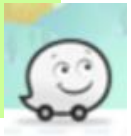


définition d'un pointeur sur le fichier

```
FILE* fichier;
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.4 - Les Fichiers : Ouverture/Fermeture



FILE* fopen(char* nom,char* mode)

- modes texte : “r“, “w“, “w+“, “r+“, “a+“
- modes binaire : “rb“, “wb“, “wb+“, “rb+“, “ab+“
- ouverture = positionnement début de fichier (sauf pour “a+“ et “ab+“
- Exemple :

```
FILE * fichier=fopen(“toto.txt“,  “r“);
```

int fclose(FILE* fichier)

- Retourne 0 si la fermeture s’est bien passée
- Exemple :

```
fclose(fichier);
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.4 - Les Fichiers : Ouverture/Fermeture

```c
/* fopen example */
#include <stdio.h>
int main ()
{
  FILE * pFile;
  pFile = fopen ("myfile.txt","w");
  if (pFile!=NULL)
  {
    fputs ("fopen example",pFile);
    fclose (pFile);
  }
  return 0;
}
```

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

int main( int argc, char * argv[] ) {

    FILE * inputFile;

    argc--;   argv++;

    if ( argc == 0 ) {
        printf( "Usage: sample filename...\n" );
        exit( 0 );
    }

    inputFile = fopen( argv[0], "a" );
    if ( inputFile == NULL ) {
        printf( "Cannot open file %s\n", argv[0] );
        exit( 0 );
    }

    fprintf( inputFile, "Appending a new message in the file\n" );

    fclose( inputFile );

    return 0;
}
```

Réf : https://koor.fr/C/Index.wp https://www.cplusplus.com/reference/

V4,0 - Copyright : Yantra Technologies 2004-2015

13.6 - Les Fichiers : Destruction

int remove(char* nom)

- Retourne 0 si la destruction s'est bien passée
- Exemple :

remove(“toto.txt“);

```
/* remove example: remove myfile.txt */
#include <stdio.h>

int main ()
{
  if( remove( "myfile.txt" ) != 0 )
    perror( "Error deleting file" );
  else
    puts( "File successfully deleted" );
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.7 - Les Fichiers : Renommer

int rename(char* ancien_nom,char* nouv_nom)

- Retourne 0 si l'opération s'est bien passée
- Exemple :

rename("toto.txt", "tutu.txt");

```
/* rename example */
#include <stdio.h>

int main ()
{
  int result;
  char oldname[] ="oldname.txt";
  char newname[] ="newname.txt";
  result= rename( oldname , newname );
  if ( result == 0 )
    puts ( "File successfully renamed" );
  else
    perror( "Error renaming file" );
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.8 - Les Fichiers : Fichiers binaires -> Ecriture

int fwrite(void* p,int taille_bloc,int nb_bloc,FILE* fic)

- Ecrit les nb_blocs de taille taille_bloc pointés par p dans le fichier fic
- Exemple : int tab[3];

  fwrite(tab,sizeof(int),3,fichier);

```
/* fwrite example : write buffer */
#include <stdio.h>

int main ()
{
  FILE * pFile;
  char buffer[] = { 'x' , 'y' , 'z' };
  pFile = fopen ("myfile.bin", "wb");
  fwrite (buffer , sizeof(char), sizeof(buffer), pFile).
  fclose (pFile);
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.8 - Les Fichiers : Fichiers binaires -> Lecture

int fread(void* p,int taille_bloc,int nb_bloc, FILE* fic)

- Lit les nb_blocs de taille taille_bloc pointés par p dans le fichier fic
- retourne le nombre de blocs lus et 0 si fin de fichier
- Exemple : int tab[3];

```
fread(tab,sizeof(int),3,fichier);
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.9 - Les Fichiers : Fichiers binaires -> Lecture

```
/* fread example: read an entire file */
#include <stdio.h>
#include <stdlib.h>

int main () {
  FILE * pFile;
  long lSize;
  char * buffer;
  size_t result;

  pFile = fopen ( "myfile.bin" , "rb" );
  if (pFile==NULL) {fputs ("File error",stderr); exit (1);}

  // obtain file size:
  fseek (pFile , 0 , SEEK_END);
  lSize = ftell (pFile);
  rewind (pFile);

  // allocate memory to contain the whole file:
  buffer = (char*) malloc (sizeof(char)*lSize);
  if (buffer == NULL) {fputs ("Memory error",stderr); exit (2);}

  // copy the file into the buffer:
  result = fread (buffer,1,lSize,pFile);
  if (result != lSize) {fputs ("Reading error",stderr); exit (3);

  /* the whole file is now loaded in the memory buffer. */

  // terminate
  fclose (pFile);
  free (buffer);
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.9 - Les Fichiers : Fichiers textes -> Ecriture

int fprintf(FILE* fic,char* chaîne_format,liste de données)

- Ecrit la chaîne_format (avec les données) dans le fichier fic
- Exemple : fprintf(fichier,‘‘Salut‘‘);

  fprintf(fichier, ‘‘coucou %d‘‘,i);

```
/* fprintf example */
#include <stdio.h>

int main ()
{
   FILE * pFile;
   int n;
   char name [100];

   pFile = fopen ("myfile.txt","w");
   for (n=0 ; n<3 ; n++)
   {
     puts ("please, enter a name: ");
     gets (name);
     fprintf (pFile, "Name %d [%-10.10s]\n",n+1,name);
   }
   fclose (pFile);

   return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.9 - Les Fichiers : Fichiers textes -> Lecture

int fscanf(FILE* fic,char* format,liste d'adresses)

- Lit les données en suivant le format dans le fichier fic
- Exemple : fscanf(fichier,"%d",&i);

```
/* fscanf example */
#include <stdio.h>

int main ()
{
  char str [80];
  float f;
  FILE * pFile;

  pFile = fopen ("myfile.txt","w+");
  fprintf (pFile, "%f %s", 3.1416, "PI");
  rewind (pFile);
  fscanf (pFile, "%f", &f);
  fscanf (pFile, "%s", str);
  fclose (pFile);
  printf ("I have read: %f and %s \n",f,str);
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

13.10 - Les Fichiers : Gestion des erreurs

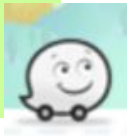
int feof (FILE * stream);

- fopen retourne NULL si erreur
- int feof(FILE* fichier) == 0 tant que la fin de fichier n'est pas atteinte

```
/* feof example: byte counter */
#include <stdio.h>

int main ()
{
  FILE * pFile;
  int n = 0;
  pFile = fopen ("myfile.txt","rb");
  if (pFile==NULL) perror ("Error opening file");
  else
  {
    while (fgetc(pFile) != EOF) {
      ++n;
    }
    if (feof(pFile)) {
      puts ("End-of-File reached.");
      printf ("Total number of bytes read: %d\n", n);
    }
    else puts ("End-of-File was not reached.");
    fclose (pFile);
  }
  return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

14 - Utilisation d'une bibliothèque



14.1 - Notion de prototype

14.2 - Fonction ne renvoyant rien

14.3 - Fonction renvoyant une valeur

14.4 - Fonction avec passage de paramètres

V4,0 - Copyright : Yantra Technologies 2004-2015

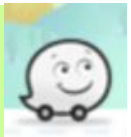
14.1 - Notion de prototype

- Fichiers .h
- Fichiers d'en-tête
- prototypes de fonctions
- Vérification de syntaxe

- exemple : prototype de getchar est int getchar()

V4,0 - Copyright : Yantra Technologies 2004-2015

14.2 - Fonction ne renvoyant rien



- Fonctions de type void
- On l ’appelle tel quel
  - exemple :
    - perror
    - prototype : void perror(const char*);
    - prototype dans stdio.h

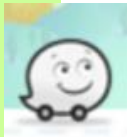
V4,0 - Copyright : Yantra Technologies 2004-2015

14.3 - Fonction renvoyant une valeur

- Fonctions de type int ou char, ou long, etc.
- On récupère sa valeur dans une variable
  - exemple :
    - getchar
    - prototype : int getchar();
    - prototype dans stdio.h

V4,0 - Copyright : Yantra Technologies 2004-2015

14.4 - Fonction avec passage de paramètres



- Fonctions utilisant le paramètre pour fonctionner
- Utilisation d'une même fonction pour plusieurs données
  - exemple :
    - log
    - prototype : double log(double);
    - prototype dans math.h

V4,0 - Copyright : Yantra Technologies 2004-2015

15 - librairies standards

15.1 - Entrées-sorties <stdio.h>

15.2 - Manipulation de caractères <ctype.h>

15.3 - Manipulation de chaînes de caractères <string.h>

15.4 - Fonctions mathématiques < math.h>

15.5 - Utilitaires divers <stdlib.h>

15.6 - Date et heure < times.h>

V4,0 - Copyright : Yantra Technologies 2004-2015

15.1a - Entrées-sorties <stdio.h>

Manipulation de fichiers	
fonction	**action**
fopen	ouverture d'un fichier
fclose	fermeture d'un fichier
fflush	écriture des buffers en mémoire dans le fichier

Entrées et sorties formatées		
fonction	**prototype**	**action**
fprintf	int fprintf(FILE *stream, char *format, ...)	écriture sur un fichier
fscanf	int fscanf(FILE *stream, char *format, ...)	lecture depuis un fichier
printf	int printf(char *format, ...)	écriture sur la sortie standard
scanf	int scanf(char *format, ...)	lecture depuis l'entrée standard
sprintf	int sprintf(char *s, char *format, ...)	écriture dans la chaîne de caractères s
sscanf	int sscanf(char *s, char *format, ...)	lecture depuis la chaîne de caractères s

V4,0 - Copyright : Yantra Technologies 2004-2015

15.1b - Entrées-sorties <stdio.h>

Impression et lecture de caractères		
fonction	**prototype**	**action**
fgetc	int fgetc(FILE *stream)	lecture d'un caractère depuis un fichier
fputc	int fputc(int c, FILE *stream)	écriture d'un caractère sur un fichier
getc	int getc(FILE *stream)	équivalent de fgetc mais implémenté par une macro
putc	int putc(int c, FILE *stream)	équivalent de fputc mais implémenté par une macro
getchar	int getchar(void)	lecture d'un caractère depuis l'entrée standard
putchar	int putchar(int c)	écriture d'un caractère sur la sortie standard
fgets	char *fgets(char *s, FILE *stream)	lecture d'une chaîne de caractères depuis un fichier
fputs	int *fputs(char *s, FILE *stream)	écriture d'une chaîne de caractères sur un fichier
gets	char *gets(char *s)	lecture d'une chaîne de caractères sur l'entrée standard
puts	int *puts(char *s)	écriture d'une chaîne de caractères sur la sortie standard

V4,0 - Copyright : Yantra Technologies 2004-2015

15.2 - Manipulation de caractères <ctype.h>

fonction	renvoie 1 si le caractère est
isalnum	une lettre ou un chiffre
isalpha	une lettre
iscntrl	un caractère de commande
isdigit	un chiffre décimal
isgraph	un caractère imprimable ou le blanc
islower	une lettre minuscule
isprint	un caractère imprimable (pas le blanc)
ispunct	un caractère imprimable qui n'est ni une lettre ni un chiffre
isspace	un blanc
isupper	une lettre majuscule
isxdigit	un chiffre hexadécimal

V4,0 - Copyright : Yantra Technologies 2004-2015

15.3 - Manipulation de chaînes de caractères <string.h>

fonction	prototype	action
strcpy	char *strcpy(char *ch1, char *ch2)	copie la chaîne ch2 dans la chaîne ch1 ; retourne ch1.
strncpy	char *strcpy(char *ch1, char *ch2, int n)	copie n caractères de la chaîne ch2 dans la chaîne ch1 ; retourne ch1.
strcat	char *strcat(char *ch1, char *ch2)	copie la chaîne ch2 à la fin de la chaîne ch1 ; retourne ch1.
strncat	char *strncat(char *ch1, char *ch2, int n)	copie n caractères de la chaîne ch2 à la fin de la chaîne ch1 ; retourne ch1.
strcmp	int strcmp(char *ch1, char *ch2)	compare ch1 et ch2 pour l'ordre lexicographique ; retourne une valeur négative si ch1 est inférieure à ch2, une valeur positive si ch1 est supérieure à ch2, 0 si elles sont identiques.
strncmp	int strcmp(char *ch1, char *ch2, int n)	compare les n premiers caractères de ch1 et ch2.
strchr	char *strchr(char *chaine, char c)	retourne un pointeur sur la première occurence de c dans chaine, et NULL si c n'y figure pas.
strrchr	char *strchr(char *chaine, char c)	retourne un pointeur sur la dernière occurence de c dans chaine, et NULL si c n'y figure pas.
strstr	char *strchr(char *ch1, char *ch2)	retourne un pointeur sur la première occurence de ch2 dans ch1, et NULL si ch2 n'y figure pas.
strlen	int strlen(char *chaine)	retourne la longueur de chaine.

V4,0 - Copyright : Yantra Technologies 2004-2015

15.4 - Fonctions mathématiques <math.h>

fonction	action
acos	arc cosinus
asin	arc sinus
atan	arc tangente
cos	cosinus
sin	sinus
tan	tangente
cosh	cosinus hyperbolique
sinh	cosinus hyperbolique
tanh	tangente hyperbolique
exp	exponentielle
log	logarithme népérien
log10	logarithme en base 10
pow	puissance
sqrt	racine carrée
fabs	valeur absolue
fmod	reste de la division
ceil	partie entière supérieure
floor	partie entière inférieure

V4,0 - Copyright : Yantra Technologies 2004-2015

15.5a - Utilitaires divers <stdlib.h>

Allocation dynamique	
fonction	**action**
calloc	allocation dynamique et initialisation à zéro.
malloc	allocation dynamique
realloc	modifie la taille d'une zone préalablement allouée par calloc ou malloc.
free	libère une zone mémoire

Conversion de chaînes de caractères en nombres		
fonction	**prototype**	**action**
atof	double atof(char *chaine)	convertit chaine en un double
atoi	int atoi(char *chaine)	convertit chaine en un int
atol	long atol(char *chaine)	convertit chaine en un long int

V4,0 - Copyright : Yantra Technologies 2004-2015

15.5b - Utilitaires divers <stdlib.h>

Génération de nombres pseudo-aléatoires		
fonction	**prototype**	**action**
rand	int rand(void)	fournit un nombre entier pseudo-aléatoire
srand	void srand(unsigned int germe)	modifie la valeur de l'initialisation du générateur pseudo-aléatoire utilisé par rand.

Arithmétique sur les entiers		
fonction	**prototype**	**action**
abs	int abs(int n)	valeur absolue d'un entier
labs	long labs(long n)	valeur absolue d'un long int
div	div_t div(int a, int b)	quotient et reste de la division euclidienne de a par b. Les résultats sont stockés dans les champs quot et rem (de type int) d'une structure de type div_t.
ldiv	ldiv_t ldiv(long a, long b)	quotient et reste de la division euclidienne de a par b. Les résultats sont stockés dans les champs quot et rem (de type long int) d'une structure de type ldiv_t.

V4,0 - Copyright : Yantra Technologies 2004-2015

15.5c - Utilitaires divers <stdlib.h>

Recherche et tri	
fonction	**action**
qsort	permet de trier un tableau
bsearch	permet de rechercher un élément dans un tableau déjà trié

Communication avec l'environnement		
fonction	**prototype**	**action**
abort	void abort(void)	terminaison anormale du programme
exit	void exit(int etat)	terminaison du programme ; rend le contrôle au système en lui fournissant la valeur etat (la valeur 0 est considérée comme une fin normale).
system	int system(char *s)	exécution de la commande système définie par la chaîne de caractères s.

V4,0 - Copyright : Yantra Technologies 2004-2015

15.6 - Date et heure <time.h>

fonction	prototype	action
time	time_t time(time_t *tp)	retourne le nombre de secondes écoulées depuis le 1er janvier 1970, 0 heures G.M.T. La valeur retournée est assignée à *tp.
difftime	double difftime(time_t t1, time_t t2)	retourne la différence t1 - t2 en secondes.
ctime	char *ctime(time_t *tp)	convertit le temps système *tp en une chaîne de caractères explicitant la date et l'heure sous un format prédéterminé.
clock	clock_t clock(void)	retourne le temps CPU en microsecondes utilisé depuis le dernier appel à clock.

V4,0 - Copyright : Yantra Technologies 2004-2015

15.7a – C librairies

- <cassert> (assert.h)
- <cctype> (ctype.h)
- <cerrno> (errno.h)
- <cfloat> (float.h)
- <ciso646> (iso646.h)
- <climits> (limits.h)
- <clocale> (locale.h)
- <cmath> (math.h)
- <csetjmp> (setjmp.h)
- <csignal> (signal.h)
- <cstdarg> (stdarg.h)
- <cstdbool> (stdbool.h) C++11
- <cstddef> (stddef.h)
- <cstdint> (stdint.h) C++11
- <cstdio> (stdio.h)
- <cstdlib> (stdlib.h)
- <cstring> (string.h)
- <ctime> (time.h)
- <cuchar> (uchar.h) C++11
- <cwchar> (wchar.h) C++11
- <cwctype> (wctype.h) C++11

http://www.cplusplus.com/reference/new/

V4,0 - Copyright : Yantra Technologies 2004-2015

15.7b – C librairies : <cwctype> (wctype.h)

Functions

Character classification functions

They check whether the character passed as parameter belongs to a certain category:

iswalnum	Check if wide character is alphanumeric (function)
iswalpha	Check if wide character is alphabetic (function)
iswblank	Check if wide character is blank (function)
iswcntrl	Check if wide character is a control character (function)
iswdigit	Check if wide character is decimal digit (function)
iswgraph	Check if wide character has graphical representation (function)
iswlower	Check if wide character is lowercase letter (function)
iswprint	Check if wide character is printable (function)
iswpunct	Check if wide character is punctuation character (function)
iswspace	Check if wide character is a white-space (function)
iswupper	Check if wide character is uppercase letter (function)
iswxdigit	Check if wide character is hexadecimal digit (function)

http://www.cplusplus.com/reference/new/

V4,0 - Copyright : Yantra Technologies 2004-2015

15.7c – C librairies : <cwctype> (wctype.h)

Character conversion functions

Two functions that convert between letter cases:

towlower	Convert uppercase wide character to lowercase (function)
towupper	Convert lowercase wide character to uppercase (function)

Extensible classification/conversion functions

iswctype	Check if wide character has property (function)
towctrans	Convert using transformation (function)
wctrans	Return character transformation (function)
wctype	Return character property (function)

Types

wctrans_t	Wide character transformation (type)
wctype_t	Wide character type (type)
wint_t	Wide character integral type (type)

Constants

WEOF	Wide End-of-File (constant)

http://www.cplusplus.com/reference/new/

V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C99

- Type de retour obligatoire
- Déclaration implicite de fonction est impossible
- Return sans expression dans une fonction qui fourni un résultat n'est plus autorisé
- La division / et le modulo % sont normalisés pour les nombres négatifs
  - -11/3 vaut -3
  - 11/-3 vaut -3
  - -11/-3 vaut 3
  - -11%3 vaut -2
  - 11%-3 vaut 2
  - -11%-3 vaut -2
- Emplacement des déclarions plus de contraintes
- Commentaire en fin de ligne // accepter comme en C++ et java

V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C99

- Tableau de dimension variable est accepté

```
static void vlaPrint(int taille1, int taille2, int table[taille1][taille2])
{
    // Ecriture du tableau sur stdout
    for (int i=0; i< taille1; i++)
    {
        for (int j=0; j< taille2; j++)
        {
            (void)printf("table[%d][%d] = %d\n", i, j, table[i][j]);
        }
    }
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C99

- Nouveaux types
  - long long
  - Type entier étendue :
    - int8, int16, int32, int64
    - uint8, uint16, uint32, uint64
  - Type flottant : double_ et float_t (désigne le type plus rapide et égal ou supérieur en taille)
  - Type boolean : bool
  - Type complexe

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

int main() {

    double complex x = 1 + 2*I;
    double complex y = 2 - 3*I;

    double complex z = x + y;;

    printf("z == %.1f%+.1fi\n", creal(z), cimag(z))

    return EXIT_SUCCESS;
}
```

- Nouvelles fonctions mathématiques

V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C99

- Fonction en line inline
- Caractère étendus (C99) et unicode (C11)
- Pointeur Restrint : restrict
  - restrict dit que deux pointeurs ne peuvent pas pointer vers des régions de mémoire se chevauchant. L'usage le plus courant est pour les arguments de fonction.
- Les directives #pragma qui sont spécifiques à la machine ou au système
- Les calculs flottants
  - Norme IEEE754
  - Choix du mode d'arrondi
  - Gestion des situations d'exception fenv,h

V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C99

- Fonction en line inline
- Caractère étendus (C99) et unicode (C11)
- Pointeur Restrint : restrict
  - restrict dit que deux pointeurs ne peuvent pas pointer vers des régions de mémoire se chevauchant. L'usage le plus courant est pour les arguments de fonction.
- Les directives #pragma qui sont spécifiques à la machine ou au système
- Les calculs flottants
  - Norme IEEE754
  - Choix du mode d'arrondi
  - Gestion des situations d'exception fenv,h

FE_DIVBYZERO	Pole error exception (macro)
FE_INEXACT	Inexact result exception (macro)
FE_INVALID	Invalid argument exception (macro)
FE_OVERFLOW	Overflow range error exception (macro)
FE_UNDERFLOW	Underflow range error exception (macro)
FE_ALL_EXCEPT	All exceptions (macro)

V4,0 - Copyright : Yantra Technologies 2004-2015

16 - Evolution C99 et C11

Depuis C99

- Structure incomplète

```c
struct token_t
{
    char* start;
    int length;
    int type;
};
```

- Structure anonyme (C11)

```c
// anonymous_structures.c
#include <stdio.h>

struct phone
{
    int  areacode;
    long number;
};

struct person
{
    char   name[30];
    char   gender;
    int    age;
    int    weight;
    struct phone;    // Anonymous structure; no name needed
} Jim;

int main()
{
    Jim.number = 1234567;
    printf_s("%d\n", Jim.number);
}
//Output: 1234567
```


V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C11

- Expressions fonctionnelles génériques _Generic
  - Les sélections génériques permettent de construire des expressions en fonction du type d'une autre expression. (mélange entre l'opérateur conditionnel et le switch)

```c
#include <stdio.h>

int main(void)
{
    printf("%d\n",_Generic( 1.0L, float:1, double:2, long double:3, default:0));
    printf("%d\n",_Generic( 1L, float:1, double:2, long double:3, default:0));
    printf("%d\n",_Generic( 1.0L, float:1, double:2, long double:3));
}
```

```
3
0
3
```


V4,0 - Copyright : Yantra Technologies 2004-2015

16 - Evolution C99 et C11

Depuis C11

- Gestion des contrainte d'alignement
  _Alignas()
  alignas()
  alignof()
  _Alignof

spécifier l'alignement précis des objets en mémoire pour tirer le meilleur parti de l'architecture matérielle.

```c
#include <stdalign.h>
#include <stdio.h>
// chaque objet de type struct sse_t sera aligné sur une limite de 16 octets
// (remarque : nécessite un support pour DR 444)
struct sse_t
{
  alignas(16) float sse_data[4];
};

// chaque objet de type struct data sera aligné sur une limite de 128 octets
struct data {
  char x;
  alignas(128) char cacheline[128]; // tableau de caractères sur-aligné,
                                    // pas un tableau de caractères sur-alignés
};

int main(void)
{
    printf("sizeof(data) = %zu (1 byte + 127 bytes padding + 128-byte array)\n",
           sizeof(struct data));

    printf("alignment of sse_t is %zu\n", alignof(struct sse_t));

    alignas(2048) struct data d; // cette instance de données est alignée encore plus stricte
}
```

https://runebook.dev/fr/docs/c/language/alignas

```
sizeof(data) = 256 (1 byte + 127 bytes padding + 128-byte array)
alignment of sse_t is 16
```


V4,0 - Copyright : Yantra Technologies 2004-2015

Depuis C11

- Fonction vérifiant le débordement mémoire : strcpy_s, strcat_s, memcpy_s, memmove_s

- Les threads : thread.h

- _Noreturn : indique au compilateur que la fonction à laquelle elle est appliquée ne retourne pas à l'appelant.

- _Static_assert() ou static_assert() Teste une assertion au moment de la compilation. Si l'expression constante spécifiée est false , le compilateur affiche le message spécifié et la compilation échoue avec l'erreur C2338 ; sinon, il n'y a aucun effet

- fopen option x ouverture exclusive du fichier

V4,0 - Copyright : Yantra Technologies 2004-2015

16 - Evolution C99 et C11

Depuis C11

- Expressions fonctionnelles génériques _Generic
- Gestion des contraintes d'alignement : _Alignas()
- Fonction vérifiant le debordement mémoire : strcpy_s, strcat_s, memcpy_s, memmove_s
- Les threads : thread.h
- _Noreturn
- _Static_assert
- fopen option x ouverture exclusive du fichier

V4,0 - Copyright : Yantra Technologies 2004-2015

V4,0 - Copyright : Yantra Technologies 2004-2015

V4,0 - Copyright : Yantra Technologies 2004-2015

17 – SDL Simple DirectMedia Layer

Simple DirectMedia Layer est une bibliothèque logicielle libre. Son API est utilisée pour créer des applications multimédias en deux dimensions pouvant comprendre du son comme les jeux vidéo, les démos graphiques, les émulateurs, etc …

https://www.libsdl.org/

Get the current **stable** SDL version 2.0.3

SDL Simple Directmedia Layer

Main
- About
- Bugs
- Licensing
- Credits
- Feedback

Documentation
- Wiki
- Forums
- Mailing Lists

Download
- SDL 2.0
- SDL 1.2
- SDL Mercurial
- Bindings

About SDL

Simple DirectMedia Layer is a cross-platform development library designed to provide low level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D. It is used by video playback software, emulators, and popular games including Valve's award winning catalog and many Humble Bundle games.

SDL officially supports Windows, Mac OS X, Linux, iOS, and Android. Support for other platforms may be found in the source code.

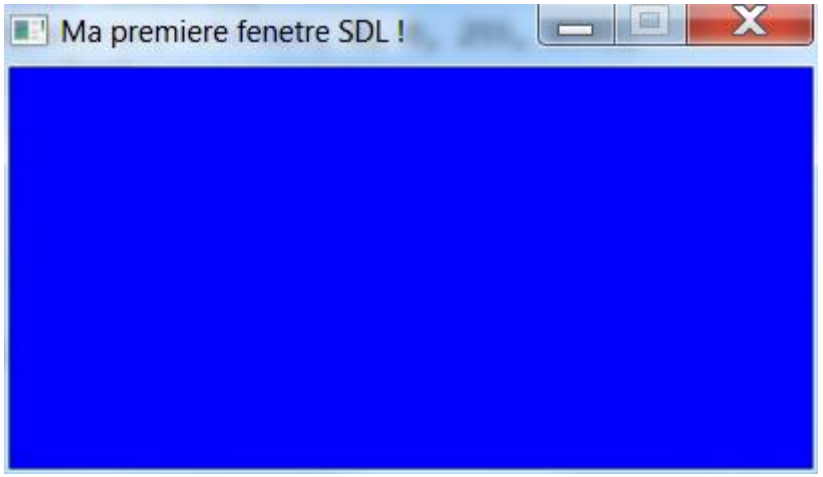
SDL is written in C, works natively with C++, and there are bindings available for several other languages, including C# and Python.

SDL 2.0 is distributed under the zlib license. This license allows you to use SDL freely in any software.

Made with SDL: Musaic Box

V4,0 - Copyright : Yantra Technologies 2004-2015

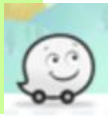
17.1a – SDL : Premier programme graphique



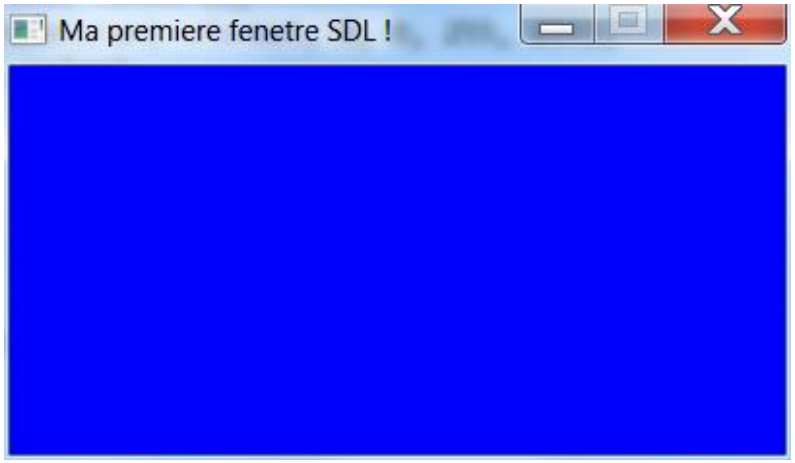

```
int main(int argc, char* args[]) {
    SDL_Event evt;
    SDL_Window* fenetre = NULL;
    SDL_Surface* surfaceEcran = NULL;

    if (!SDL_creerWindow(&fenetre, &surfaceEcran)) {
        printf("Erreur de creation!\n");
    } else {
        SDL_UpdateWindowSurface(fenetre);
        while (1) {
            //Handle events on queue
            if (SDL_PollEvent(&evt) != 0) {
                //User requests quit
                if (evt.type == SDL_QUIT)
                    break;
            }
        }
    }
    SDL_libererMemoire(&fenetre);
    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.1b – SDL : Premier programme graphique

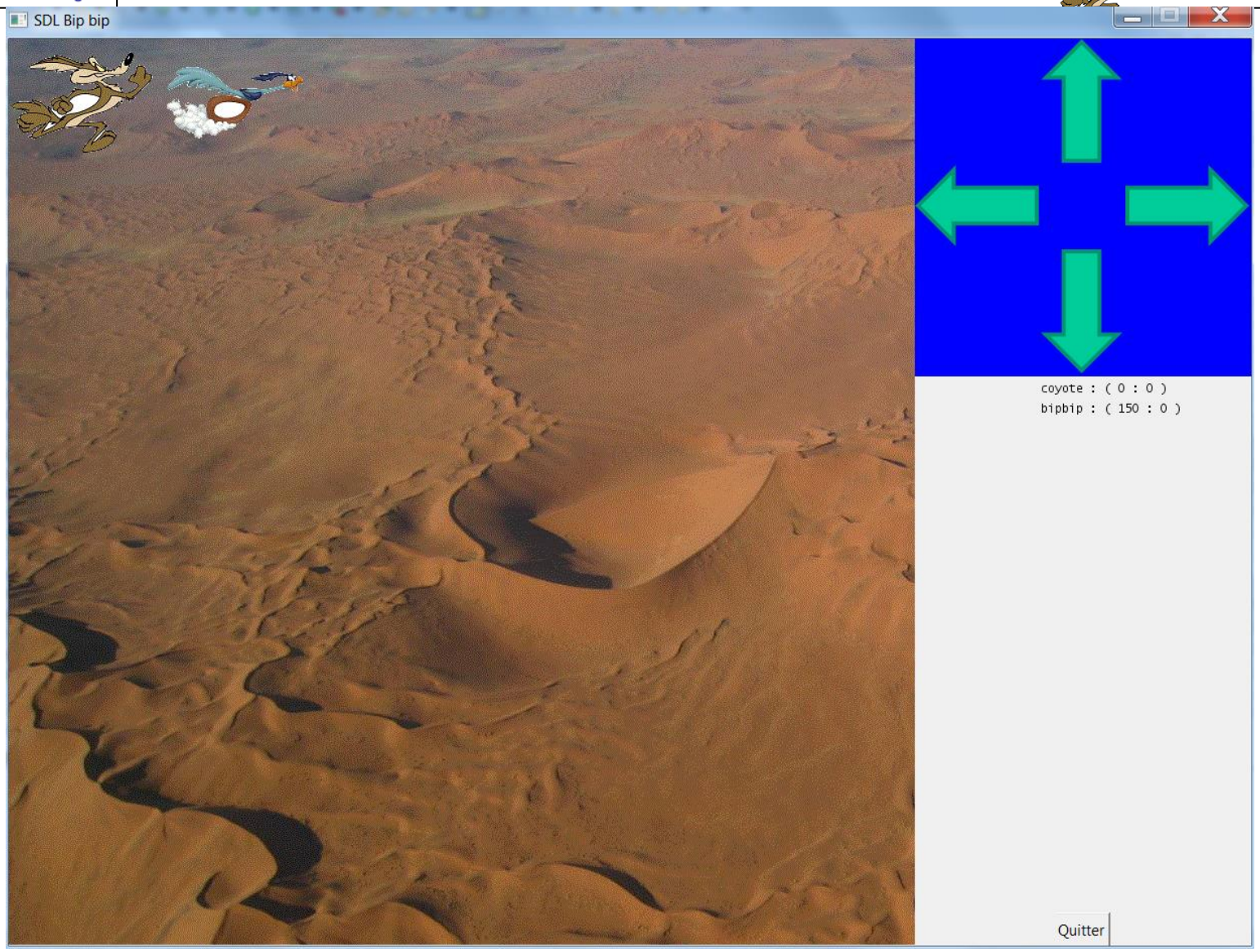


```
#include <SDL.h>
#include <stdio.h>
#include <stdlib.h>

bool SDL_creerWindow(SDL_Window** fenetre, SDL_Surface** surfaceEcran) {
    if (SDL_Init( SDL_INIT_VIDEO) < 0) {
        printf("SDL_Error: %s\n", SDL_GetError());
        return false;
    } else {
        *fenetre = SDL_CreateWindow("Ma premiere fenetre SDL !",SDL_WINDOWPOS_CENTERED,SDL_WINDOWPOS_CENTERED, 400, 200, 0);
        if (fenetre == NULL) {
            printf("SDL_Error: %s\n", SDL_GetError());
            return false;
        } else {
            *surfaceEcran = SDL_GetWindowSurface(*fenetre);
            Uint32 color = SDL_MapRGBA((*surfaceEcran)->format, 0, 0, 255, 255);
            SDL_FillRect(*surfaceEcran, NULL, color);
            return true;
        }
    }
    return false;
}

void SDL_libererMemoire(SDL_Window** fenetre) {
    SDL_DestroyWindow(*fenetre);
    *fenetre = NULL;
}
```



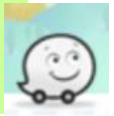
V4,0 - Copyright : Yantra Technologies 2004-2015

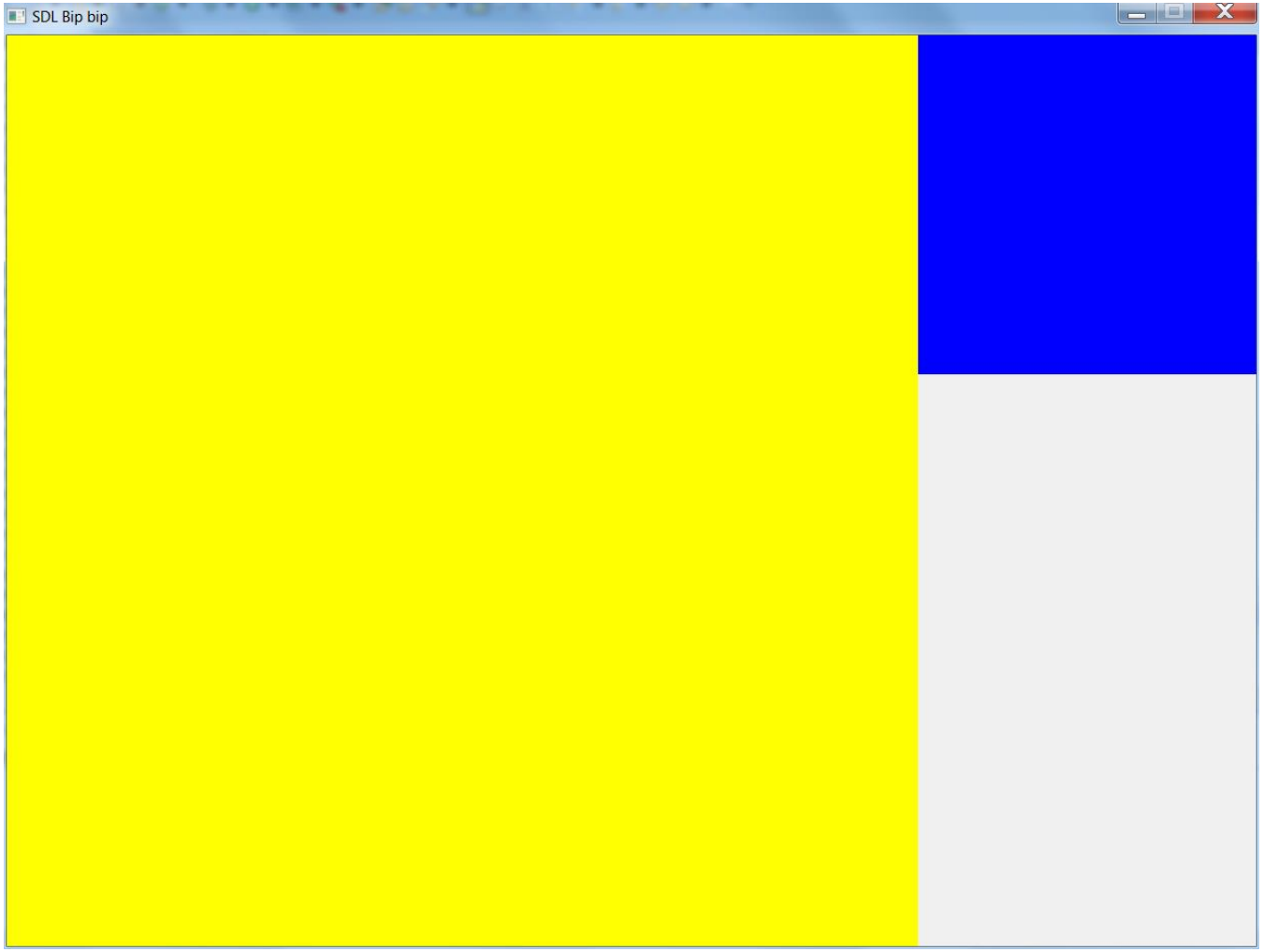
17.2 – SDL : Exemple -> Bip Bip et Coyote



V4,0 - Copyright : Yantra Technologies 2004-2015

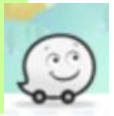
17.2 – SDL : Exemple -> Cadre





V4,0 - Copyright : Yantra Technologies 2004-2015

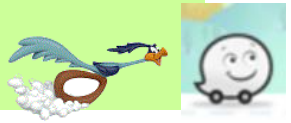
17.2 – SDL : Exemple -> Cadre



```c
int main(int argc, char* args[]) {
    BOOL quit = FALSE;
    SDL_Event evt;
    SDL_Window* fenetre = NULL;
    SDL_Surface* surfaceEcran = NULL;

    if (!SDL_creerWindow(&fenetre, &surfaceEcran)) {
        printf("Erreur d'initialisation!\n");
    } else {
        //Load media
        if (!SDL_Initialisation_Media(fenetre, surfaceEcran)) {
            printf("Erreur de lecture des media !\n");
        } else {
            SDL_UpdateWindowSurface(fenetre);
            while (!quit) {
                //Handle events on queue
                while (SDL_PollEvent(&evt) != 0) {
                    quit = SDL_evenenement(&evt);
                }
            }
        }
    }
    SDL_libererMemoire(&fenetre);
    return 0;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Cadre



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

#define TRUE 1
#define FALSE 0
typedef short unsigned int BOOL;

#define ERREUR_STL() fprintf(stderr,"SDL_Error: %s\n", SDL_GetError());exit(EXIT_FAILURE);
#define grisclaire SDL_MapRGBA((*surfaceEcran)->format, 240, 240, 240,255)
#define bleue SDL_MapRGBA((*surfaceEcran)->format, 0, 0, 255,255)
#define jaune SDL_MapRGBA((*surfaceEcran)->format, 255, 255, 0,255)

const int TAILLE = 150;
const int RMAX = 6;
const int TMAX = TAILLE * RMAX;
const int SCREEN_WIDTH = TMAX + 125 * 2 + 84;
const int SCREEN_HEIGHT = TMAX;

int xb = TAILLE;
int yb = 0;
int xc = 0;
int yc = 0;
```

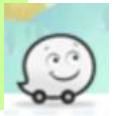

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Cadre

```
BOOL SDL_creerWindow(SDL_Window**fenetre, SDL_Surface** surfaceEcran) {
    BOOL success = TRUE;
    SDL_Rect r;

    if (SDL_Init( SDL_INIT_VIDEO | SDL_INIT_AUDIO) < 0) {
        ERREUR_STL();
    } else {
        *fenetre = SDL_CreateWindow("SDL Bip bip", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,SCREEN_WIDTH, SCREEN_HEIGHT,0 );
        if (*fenetre == NULL) {
            ERREUR_STL();
        } else {
            *surfaceEcran = SDL_GetWindowSurface(*fenetre);

            SDL_FillRect(*surfaceEcran, NULL, grisclaire);
            r.x = TMAX;
            r.y = 0;
            r.w = 125 * 2 + 85;
            r.h = 125 * 2 + 85;
            SDL_FillRect((*surfaceEcran), &r, bleue);
            r.x = 0;
            r.y = 0;
            r.w = TMAX;
            r.h = TMAX;
            SDL_FillRect((*surfaceEcran), &r, jaune);

        }
    }
    return success;
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Cadre



```
BOOL SDL_Initialisation_Media(SDL_Window*fenetre, SDL_Surface* surfaceEcran) {
    BOOL success = TRUE;
    return success;
}
```

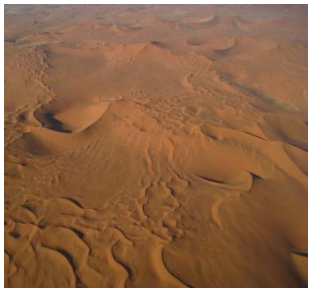
```
void SDL_libererMemoire(SDL_Window** fenetre) {
    SDL_DestroyWindow(*fenetre);
    *fenetre = NULL;
}

BOOL SDL_evenenement(SDL_Event* evt) {
    if (evt->type == SDL_QUIT)
        return TRUE;
    return FALSE;
}
```

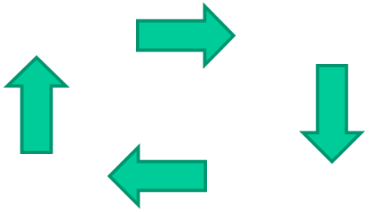
V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Définition des images

```
BOOL SDL_Initialisation_Media(SDL_Window*fenetre, SDL_Surface* surfaceEcran) {
    BOOL success = TRUE;
    SDL_Rect dstrect;
    fdesert = IMG_Load("C:\\STL_img\\desert900.png");
    fcoyote_droite = IMG_Load("C:\\STL_img\\coyote_min_droite.png");
    fbipbip_droite = IMG_Load("C:\\STL_img\\bipbip_min_droite.png");
    fcoyote_gauche = IMG_Load("C:\\STL_img\\coyote_min_gauche.png");
    fbipbip_gauche = IMG_Load("C:\\STL_img\\bipbip_min_gauche.png");
    fcoyote_haut = IMG_Load("C:\\STL_img\\coyote_min_haut.png");
    fbipbip_haut = IMG_Load("C:\\STL_img\\bipbip_min_haut.png");
    fcoyote_bas = IMG_Load("C:\\STL_img\\coyote_min_bas.png");
    fbipbip_bas = IMG_Load("C:\\STL_img\\bipbip_min_bas.png");
    fbgauche = IMG_Load("C:\\STL_img\\gauche.png");
    fbdroite = IMG_Load("C:\\STL_img\\droite.png");
    fbhaut = IMG_Load("C:\\STL_img\\haut.png");
    fbbas = IMG_Load("C:\\STL_img\\bas.png");
    if (fdesert == NULL || fcoyote_droite == NULL || fbipbip_droite == NULL
            || fcoyote_gauche == NULL || fbipbip_gauche == NULL
            || fcoyote_haut == NULL || fbipbip_haut == NULL
            || fcoyote_bas == NULL || fbipbip_bas == NULL || fbgauche == NULL
            || fbdroite == NULL || fbhaut == NULL || fbbas == NULL) {
        ERREUR_STL()
    }
    SDL_BlitSurface(fdesert, NULL, surfaceEcran, NULL);
    dstrect.x = xc;
    dstrect.y = yc;
    dstrect.w = 150;
    dstrect.h = 120;
    SDL_BlitSurface(fcoyote_droite, NULL, surfaceEcran,&dstrect);

    dstrect.x = xb;
    dstrect.y = yb;
    dstrect.w = 150;
    dstrect.h = 120;
    SDL_BlitSurface(fbipbip_droite, NULL, surfaceEcran,&dstrect);
    return success;
}
```

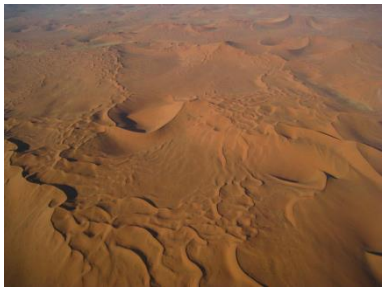
```
#include <SDL_image.h>
```



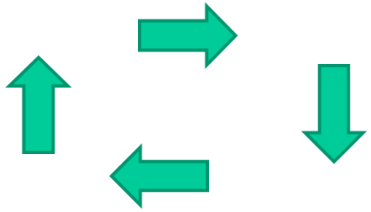


V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Définition des images



```
SDL_Surface* fdesert = NULL;
SDL_Surface* fcoyote_droite = NULL;
SDL_Surface* fbipbip_droite = NULL;
SDL_Surface* fcoyote_gauche = NULL;
SDL_Surface* fbipbip_gauche = NULL;
SDL_Surface* fcoyote_haut = NULL;
SDL_Surface* fbipbip_haut = NULL;
SDL_Surface* fcoyote_bas = NULL;
SDL_Surface* fbipbip_bas = NULL;
SDL_Surface* fbgauche = NULL;
SDL_Surface* fbdroite = NULL;
SDL_Surface* fbhaut = NULL;
SDL_Surface* fbbas = NULL;
```



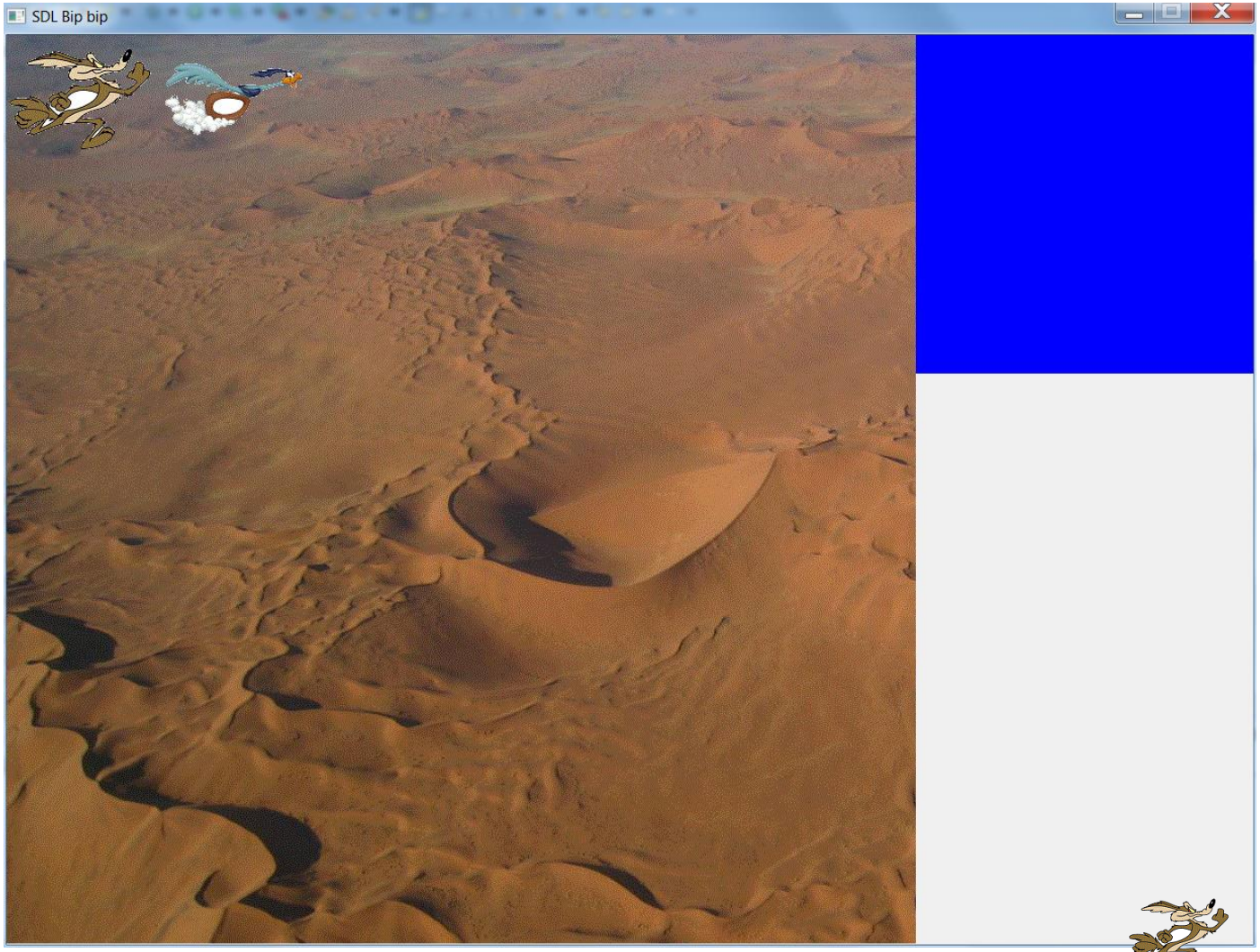
V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Définition des images

```c
void SDL_libererIMG() {
    SDL_FreeSurface(fdesert);
    SDL_FreeSurface(fcoyote_droite);
    SDL_FreeSurface(fbipbip_droite);
    SDL_FreeSurface(fcoyote_gauche);
    SDL_FreeSurface(fbipbip_gauche);
    SDL_FreeSurface(fcoyote_haut);
    SDL_FreeSurface(fbipbip_haut);
    SDL_FreeSurface(fcoyote_bas);
    SDL_FreeSurface(fbipbip_bas);
    SDL_FreeSurface(fbgauche);
    SDL_FreeSurface(fbdroite);
    SDL_FreeSurface(fbhaut);
    SDL_FreeSurface(fbbas);

    fdesert = NULL;
    fcoyote_droite = NULL;
    fbipbip_droite = NULL;
    fcoyote_gauche = NULL;
    fbipbip_gauche = NULL;
    fcoyote_haut = NULL;
    fbipbip_haut = NULL;
    fcoyote_bas = NULL;
    fbipbip_bas = NULL;
    fbgauche = NULL;
    fbdroite = NULL;
    fbhaut = NULL;
    fbbas = NULL;

}

void SDL_libererMemoire(SDL_Window** fenetre) {

    SDL_libererIMG();

    SDL_DestroyWindow(*fenetre);
    *fenetre = NULL;

    IMG_Quit();

}
```

V4,0 - Copyright : Yantra Technologies 2004-2015

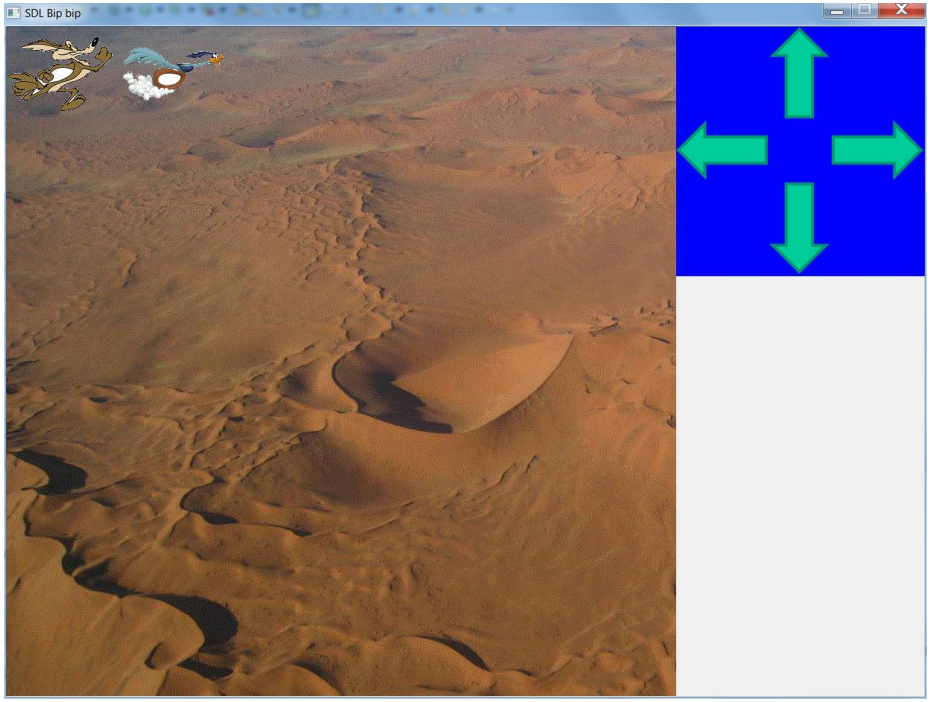
17.2 – SDL : Exemple -> Définition des images



V4,0 - Copyright : Yantra Technologies 2004-2015

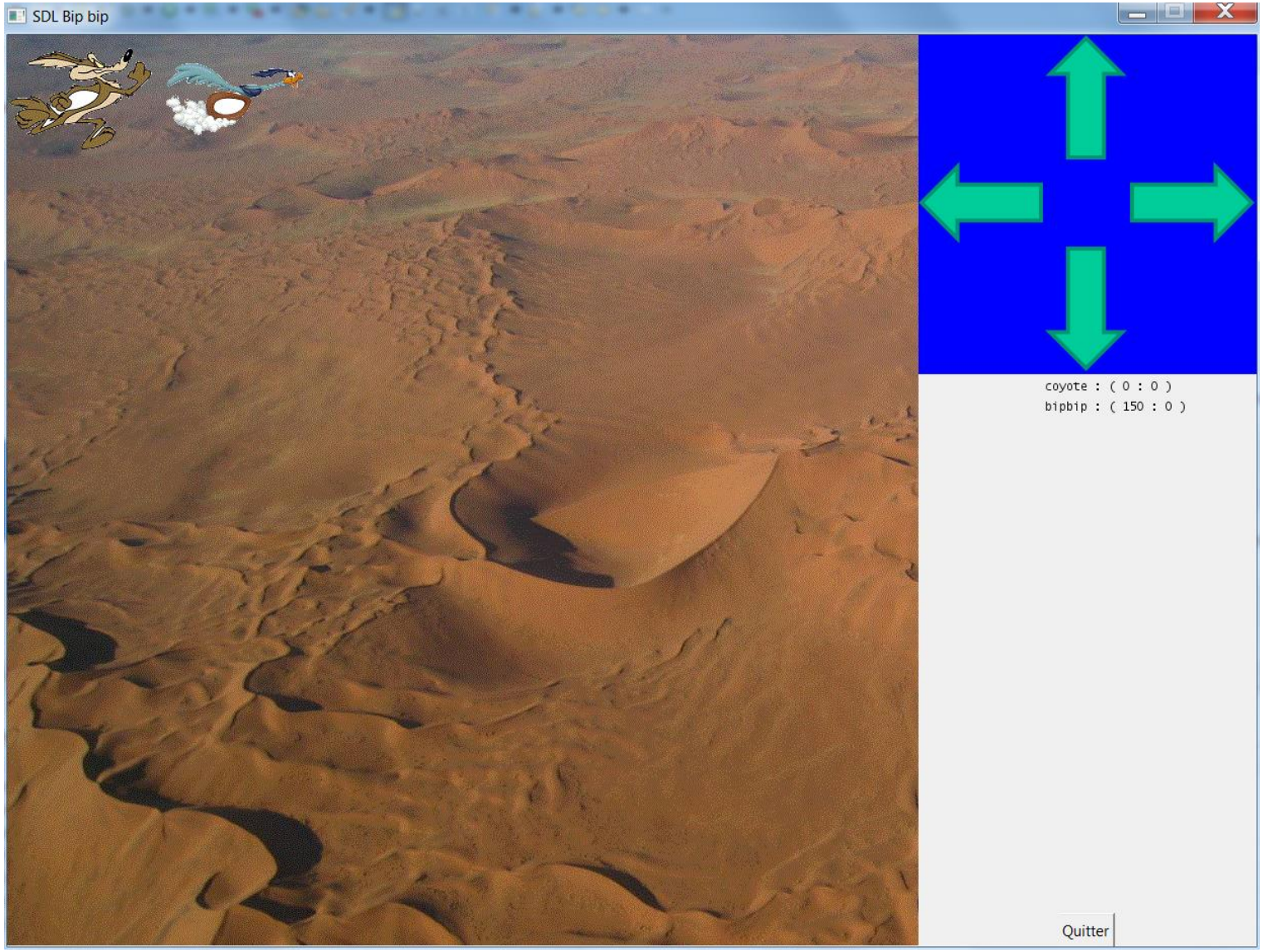
17.2 – SDL : Exemple -> Insérer définition des boutons

```
dstrect.x = xc;
dstrect.y = yc;
dstrect.w = 150;
dstrect.h = 120;
SDL_BlitSurface(fcoyote_droite, NULL, surfaceEcran,&dstrect);

dstrect.x = xb;
dstrect.y = yb;
dstrect.w = 150;
dstrect.h = 120;
SDL_BlitSurface(fbipbip_droite, NULL, surfaceEcran,&dstrect);

dstrect.x = TMAX + 125;
dstrect.y = 0;
dstrect.w = 83;
dstrect.h = 125;
SDL_BlitSurface(fbhaut, NULL, surfaceEcran, &dstrect);

dstrect.x = TMAX + 125;
dstrect.y = 125 + 84;
dstrect.w = 83;
dstrect.h = 125;
SDL_BlitSurface(fbbas, NULL, surfaceEcran, &dstrect);


dstrect.x = TMAX;
dstrect.y = 125;
dstrect.w = 125;
dstrect.h = 83;
SDL_BlitSurface(fbgauche, NULL, surfaceEcran, &dstrect);

dstrect.x = TMAX + 125 + 84;
dstrect.y = 125;
dstrect.w = 125;
dstrect.h = 83;
SDL_BlitSurface(fbdroite, NULL, surfaceEcran, &dstrect);
```



V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Insérer définition des boutons



V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Insérer définition des boutons

```
#include <SDL_ttf.h>
BOOL SDL_creerWindow(SDL_Window**fenetre, SDL_Surface** surfaceEcran) {
        if(TTF_Init()  < 0 ) { ERREUR_STL(); }
 ...................
TTF_Font *police = NULL ;

BOOL SDL_Initialisation_Media(SDL_Window*fenetre, SDL_Surface* surfaceEcran) {
    BOOL success = TRUE;
    SDL_Rect dstrect;
    SDL_Surface *texte=NULL;
    SDL_Color couleurNoire = {0, 0, 0};
    static char buffer [50];
 ...................
    sprintf(buffer,"coyote : ( %d : %d )",xc,yc);
    texte = TTF_RenderText_Blended(police,buffer, couleurNoire);
    dstrect.x =  TMAX + 125;
    dstrect.y =  125 * 2 + 84 + 5;
    SDL_BlitSurface(texte, NULL, surfaceEcran, &dstrect);
    sprintf(buffer,"bipbip : ( %d : %d )",xb,yb);
    dstrect.x =  TMAX + 125;
    dstrect.y =  125 * 2 + 84 + 5+20;
    texte = TTF_RenderText_Blended(police,buffer, couleurNoire);
    SDL_BlitSurface(texte, NULL, surfaceEcran, &dstrect);
 ...................
    SDL_Surface* fbquit = IMG_Load("C:\\STL_img\\bouton_quitter.png");
    dstrect.x = TMAX + 125 + 84 / 2 - 28;
    dstrect.y = TMAX - 33;
    SDL_BlitSurface(fbquit, NULL, surfaceEcran, &dstrect);
```

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Insérer définition des boutons

```
void SDL_libererMemoire(SDL_Window** fenetre) {

    SDL_libererIMG();

    SDL_DestroyWindow(*fenetre);
    *fenetre = NULL;

    TTF_CloseFont(police);
    TTF_Quit();
    IMG_Quit();
    SDL_Quit();
}
```

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Insérer définition des boutons

```c
int main(int argc, char* args[]) {
    BOOL quit = FALSE;
    SDL_Event evt;
    SDL_Window* fenetre = NULL;
    SDL_Surface* surfaceEcran = NULL;

    if (!SDL_creerWindow(&fenetre, &surfaceEcran)) {
        printf("Erreur d'initialisation!\n");
    } else {
        //Load media
        if (!SDL_Initialisation_Media(fenetre, surfaceEcran)) {
            printf("Erreur de lecture des media !\n");
        } else {
            SDL_UpdateWindowSurface(fenetre);
            while (!quit) {
                //Handle events on queue
                while (SDL_PollEvent(&evt) != 0) {
                    quit = SDL_evenenement(&evt,fenetre,surfaceEcran);
                }
            }
        }
    }
    SDL_libererMemoire(&fenetre);
    return 0;
}
```

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Gestion évènement : souris, clavier

```
bool testRect(int x, int y, SDL_Rect* dstrect) {
    return (x > dstrect->x) && (x < dstrect->x + dstrect->w) && (y > dstrect->y)
            && (y < dstrect->y + dstrect->h);
}
```

```
BOOL SDL_evenenement(SDL_Event* evt, SDL_Window*fenetre,SDL_Surface* surfaceEcran) {
    SDL_Rect dstrecthaut;
    SDL_Rect dstrectbas;
    SDL_Rect dstrectgauche;
    SDL_Rect dstrectdroite;
    SDL_Rect dstrectquit;
    dstrecthaut.x = TMAX + 125;dstrecthaut.y = 0;
    dstrecthaut.w = 83;dstrecthaut.h = 125;
    dstrectbas.x = TMAX + 125;dstrectbas.y = 125 + 84;
    dstrectbas.w = 83;dstrectbas.h = 125;
    dstrectgauche.x = TMAX;dstrectgauche.y = 125;
    dstrectgauche.w = 125;dstrectgauche.h = 83;
    dstrectdroite.x = TMAX + 125 + 84;dstrectdroite.y = 125;
    dstrectdroite.w = 125;dstrectdroite.h = 83;
    dstrectquit.x = TMAX + 125 + 84 / 2 - 28;dstrectquit.y = TMAX - 33;
    dstrectquit.h = 33; dstrectquit.w = 56;
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Gestion évènement : souris, clavier

```
if (evt->type == SDL_QUIT) return TRUE;
else if (evt->type == SDL_MOUSEBUTTONDOWN) {
    do {
        if (testRect(evt->motion.x, evt->motion.y, &dstrecthaut)) {
            event_haut(fenetre, surfaceEcran);
        } else if (testRect(evt->motion.x, evt->motion.y, &dstrectbas)) {
            event_bas(fenetre, surfaceEcran);
        } else if (testRect(evt->motion.x, evt->motion.y, &dstrectgauche)) {
            event_gauche(fenetre, surfaceEcran);
        } else if (testRect(evt->motion.x, evt->motion.y, &dstrectdroite)) {
            event_droite(fenetre, surfaceEcran);
        } else if (testRect(evt->motion.x, evt->motion.y, &dstrectquit)) {
            return TRUE;
            break;
        } else
            break;
        SDL_PollEvent (evt);
        if (evt->type == SDL_MOUSEBUTTONUP) break;
    } while (true);

} else if (evt->type == SDL_KEYDOWN) {
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Gestion évènement : souris, clavier

```
    } else if (evt->type == SDL_KEYDOWN) {
        do {
            switch (evt->key.keysym.sym) {
            case SDLK_UP:
                event_haut(fenetre, surfaceEcran);
                break;
            case SDLK_DOWN:
                event_bas(fenetre, surfaceEcran);
                break;
            case SDLK_LEFT:
                event_gauche(fenetre, surfaceEcran);
                break;
            case SDLK_RIGHT:
                event_droite(fenetre, surfaceEcran);
                break;
            default:
                break;
            }
            SDL_PollEvent(evt);
            if (evt->type == SDL_KEYUP) break;
        } while (true);
    }
    return FALSE;
}
```

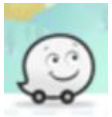

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Gestion évènement : souris, clavier

```
void event_droite(SDL_Window* screen, SDL_Surface* screenSurface) {
    SDL_Color couleurNoire = { 0, 0, 0 };
    SDL_Surface *texte = NULL;
    SDL_Rect dstrect;
    static char buffer[50];
    droite();

    SDL_BlitSurface(fdesert, NULL, screenSurface, NULL);
    dstrect.x = xc;dstrect.y = yc;
    dstrect.w = 150;dstrect.h = 120;

    SDL_BlitSurface(fcoyote_droite, NULL, screenSurface, &dstrect);
    dstrect.x = xb;dstrect.y = yb;
    dstrect.w = 150;dstrect.h = 120;

    SDL_BlitSurface(fbipbip_droite, NULL, screenSurface, &dstrect);
    dstrect.x = TMAX;dstrect.y = 125 * 2 + 85;
    dstrect.w = 125 * 2 + 85;dstrect.h = 50;

    SDL_FillRect(screenSurface, &dstrect, bleue(screenSurface));
    sprintf(buffer,"coyote : ( %d : %d )",xc,yc);
    texte = TTF_RenderText_Blended(police,buffer, couleurNoire);
    dstrect.x =  TMAX + 125;dstrect.y =  125 * 2 + 84 + 5;
    SDL_BlitSurface(texte, NULL, screenSurface, &dstrect);
    sprintf(buffer,"bipbip : ( %d : %d )",xb,yb);
    dstrect.x =  TMAX + 125;dstrect.y =  125 * 2 + 84 + 5+20;
    texte = TTF_RenderText_Blended(police,buffer, couleurNoire);
    SDL_BlitSurface(texte, NULL, screenSurface, &dstrect);
    SDL_UpdateWindowSurface(screen);
}
```

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Gestion évènement : souris, clavier



```
void droite() {
    xb = xb + TAILLE * 2;
    xc = xc + TAILLE;
    if (xb == xc and yb == yc)
        xb = xb + TAILLE;
    if (xb >= TMAX)
        xb = 0;
    if (xc >= TMAX)
        xc = 0;
    if (xb == xc and yb == yc)
        xb = xb + TAILLE;
}
```

V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Son

```
#include <SDL_mixer.h>


Mix_Chunk *son = NULL;

BOOL SDL_Initialisation_Media(SDL_Window*fenetre, SDL_Surface* surfaceEcran) {
    BOOL success = TRUE;
    SDL_Rect dstrect;
    SDL_Surface *texte = NULL;
    SDL_Color couleurNoire = { 0, 0, 0 };
    SDL_Surface* fbquit = NULL;
    static char buffer[50];

    if (Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 2048) < 0) {
            fprintf(stderr,"SDL_mixer Error: %s\n", Mix_GetError());
            return FALSE;
        }
        son = Mix_LoadWAV("C:\\STL_img\\4074.wav");
        if (son == NULL) {
            fprintf(stderr,"Failed to load beat music! SDL_mixer Error: %s\n",Mix_GetError());
            return FALSE;
        }
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.2 – SDL : Exemple -> Son

```
void event_droite(SDL_Window* screen, SDL_Surface* screenSurface) {
    SDL_Color couleurNoire = { 0, 0, 0 };
    SDL_Surface *texte = NULL;
    SDL_Rect dstrect;
    static char buffer[50];
    Mix_PlayChannel(-1, son, 0);
    droite();
     ..................
```

```
void SDL_libererMemoire(SDL_Window** fenetre) {

    SDL_libererIMG();

    SDL_DestroyWindow(*fenetre);
    *fenetre = NULL;

    Mix_FreeChunk(son);
    Mix_Quit();

    TTF_CloseFont(police);
    TTF_Quit();
    IMG_Quit();
    SDL_Quit();
}
```


V4,0 - Copyright : Yantra Technologies 2004-2015

17.3 - SDL : Lien

http://lazyfoo.net/tutorials/SDL

https://zestedesavoir.com/tutoriels/1014/utiliser-la-sdl-en-langage-c/la-sdl/#1-presentation-de-la-sdl

https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/17117-installation-de-la-sdl

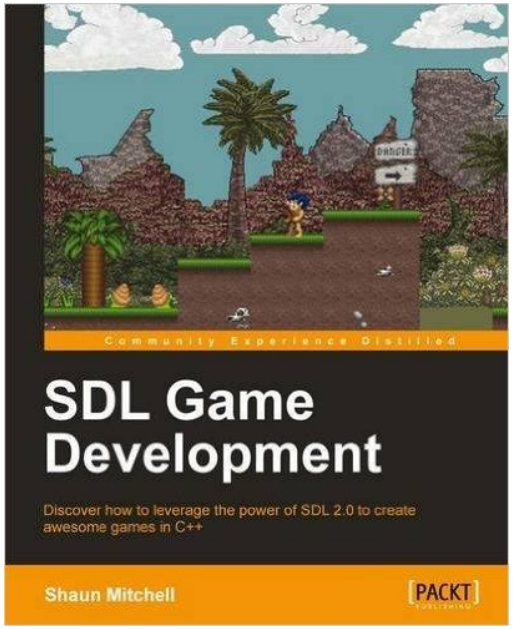
http://loka.developpez.com/tutoriel/sdl/installation/eclipse/

http://jeux.developpez.com/tutoriels/sdl-2/guide-migration/

V4,0 - Copyright : Yantra Technologies 2004-2015

17.3 - SDL : Bibliographie

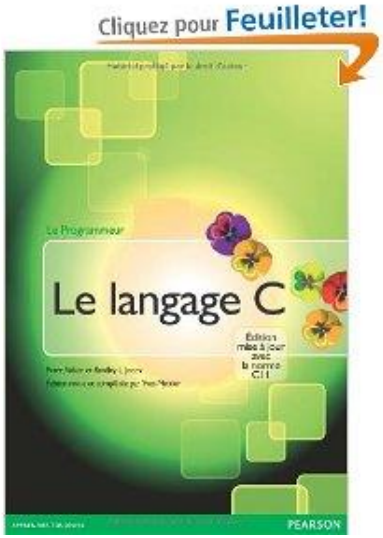
https://zestedesavoir.com/tutoriels/1014/utiliser-la-sdl-en-langage-c/

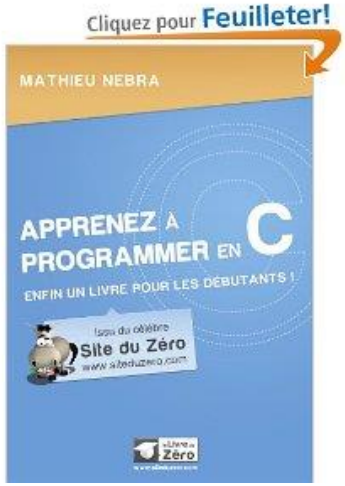


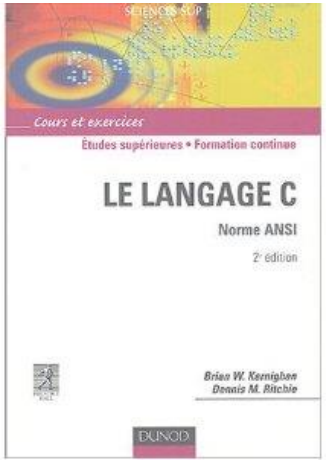
V4,0 - Copyright : Yantra Technologies 2004-2015

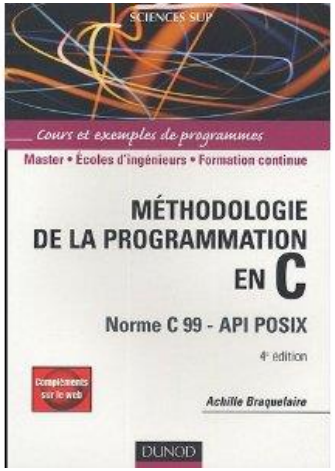
5 - Bibliographie

- **Le langage C : Norme ANSI**
  - Brian-W Kernighan et Dennis-M Ritchie
- **Langage C**
  - Claude Delannoy
- **Méthodologie de la programmation en C : Norme C 99 - API POSIX**
  - Achille Braquelaire









V4,0 - Copyright : Yantra Technologies 2004-2015

5 - Bibliographie

Quelques liens

- http://gcc.gnu.org/
- http://c.developpez.com/
- http://www.cplusplus.com/reference/clibrary/
- http://www-roc.inria.fr/secret/Anne.Canteaut/COURS_C/
- http://www.inf.enst.fr/~charon/CFacile/
- http://www.ltam.lu/Tutoriel_Ansi_C/
- http://www.siteduzero.com/tutoriel-3-14189-apprenez-a-programmer-en-c.html

V4,0 - Copyright : Yantra Technologies 2004-2015