

Programmation C

- Apprendre à développer des programmes simples dans le langage C
- Notes de cours sont disponibles sur <http://astro.u-strasbg.fr/scyon/STUSM>
(attention les majuscules sont importantes)

Modalités d'évaluation

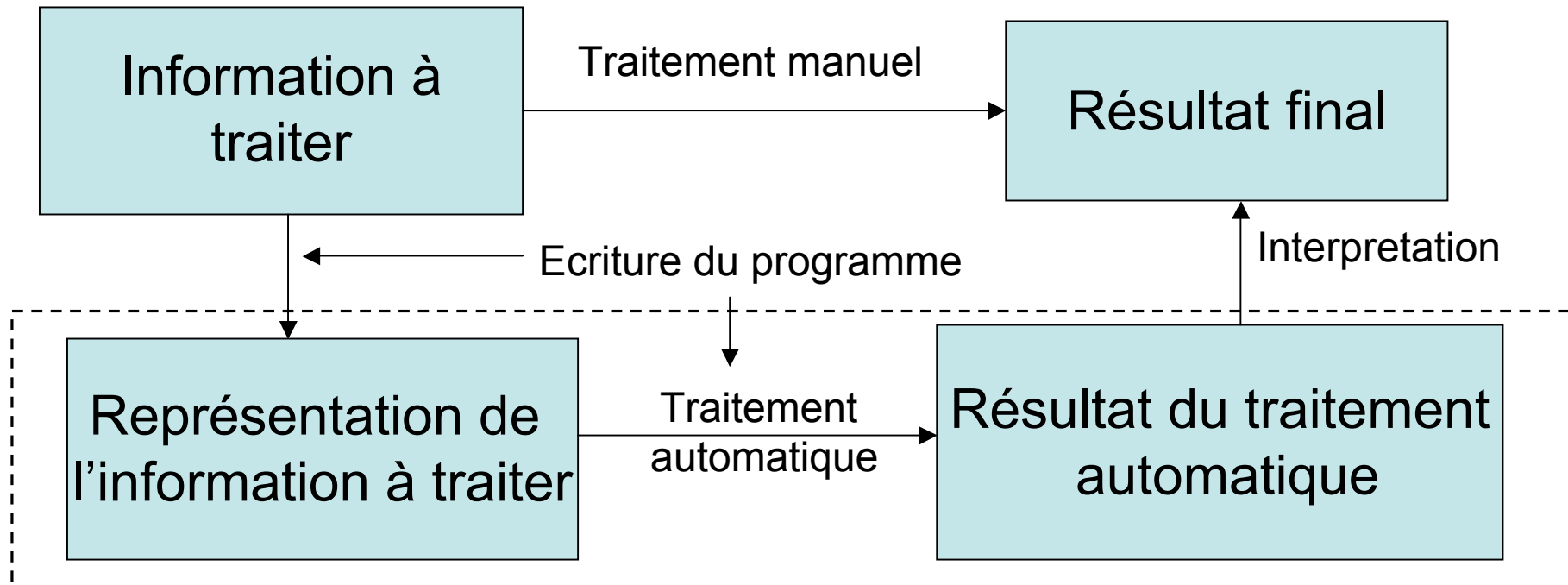
- Contrôle continu : 1/3 de la note
 - Il faut conserver les codes développés pendant les TPs
 - Les TPs sont OBLIGATOIRES!
 - 5 TPs, 2 sont notés sur les 4 derniers
 - Les comptes-rendus de TPS sont à rendre à la fin de chaque séance (à préparer donc)
- Examen final : 2/3 de la note

Que mettre dans le compte-rendu d'un TP

- Une description technique du problème:
 - Choix du type de variable
 - Choix de la méthode (quel type de boucle et pourquoi, pourquoi un si-sinon etc)
 - L'algorithme (pas le programme C)

 - Les programmes C seront à envoyer par mail à l'adresse suivante:
siebert@astro.u-strasbg.fr

- Informatique = traitement automatique de l'information
- Un ordinateur est une MACHINE qui effectue une suite d'opérations logiques
 - Stockage et traitement de l'information
 - Bête et rapide



L'ordinateur

- A l'ordre 0 un ordinateur c'est:
 - Processeur: CPU qui fait les calculs
 - Mémoire vive: RAM stocke les variables durant l'exécution d'un programme (mémoire à accès rapide)
 - Mémoire morte: ROM mémoire à accès protégé, stocke les firmware système dépendants
 - Mémoire temporaire: disque dur, stocke les données
 - Périphériques: souris, clavier, écran...

Types d'ordinateurs

- Il existe plusieurs type d'ordinateurs
 - PC: 1 ou plusieurs processeurs dans une unité centrale
 - Clusters: groupe de PC sur un même site qui peuvent fonctionner seul ou ensemble pour le calcul partagé. Chaque PC possède sa propre RAM
 - Machines parallèles: ensemble de cartes mères qui partagent 1 seule RAM
 - Grilles: machines distantes et hétérogènes

Algorithmes: bases

- Un ordinateur effectue des opérations logiques
- Une suite d'opérations logiques = algorithme (fourni par l'utilisateur)
- Une fois traduit en langage de programmation (C, Fortran, Ada etc), c'est un programme

Algorithme: bases

- Quelques règles:
 - Un ordinateur ne réfléchit pas -> incapable de corriger une erreur -> ne remplace pas la réflexion
 - Un ordinateur ne se trompe jamais (ou presque), l'erreur est toujours humaine
- Le programme est conçu par un humain, l'ordinateur l'exécute

Codage de l'information

- Un ordinateur utilise uniquement le langage binaire pour le stockage et le calcul: 0 ou 1
 - 1 bit = 0 ou 1
 - 1 octet = 8 bits = 1 byte
 - 1 mot = 8, 16, 24 ou 32 bits = 1, 2, 3 ou 4 octets
 - 1 ko = 2^{10} bits = 1024 octets; 1 Mo = 2^{20} octets; 1 Go = 2^{30} octets

Codage de l'information

- Le nombre de canaux (bits) utilisé fixe la limite de codage
 - Exemple: 1 canal= 1 bit=2 entiers possible
 - Codage sur 2 canaux= 4 positions possibles: 00,01,10,11
- Couleurs d'un écran d'ordinateur
 - Monochrome = 1 Canal=2 couleurs
 - CGA=2 bits=4 couleurs
 - EGA=4 bits=16 couleurs
 - VGA=8 bits+ 256 couleurs

Notion de codage binaire

- Binaire= base 2 (0 ou 1 seulement)

Les opérations courantes existent comme pour les autres bases et le principe est le même:

Addition: $2+3_{10}=10+11_2=101_2$

Soustraction

Multiplication

Division

Etc.

Décimal	Binaire
1	1
2	10
3	11
4	100
5	101
6	110

Notion de codage binaire

- Pour coder des chiffres négatifs on utilise le complément à 2:
 - On définit le nombre de bits pour le codage
 - On écrit le code binaire de la valeur absolue
 - On inverse la valeur des bits
 - On ajoute 1
- Ce système permet de conserver la propriété d'addition (la plus importante pour un ordinateur)

Notion de codage binaire

- Codage des nombres réels
 - Les nombres réels sont codés sur 32 ou 64 bits en fonction de l'architecture de la machine
 - La norme IEEE en 32 bits prévoit 1 bit de signe (S) suivi de 8 bits pour l'exposant (E) et 23 bits de mantisse (M)
 - le schéma de codage est donc le suivant:
 $(-1)^S M \cdot 2^{(E-127)}$

Petit exercice

- Ex 1: écrire en binaire les valeurs 307, 255, -5, -255

Combien de bits sont nécessaires?

- Ex 2: dans le codage des réels, quelle(s) est(sont) les limitations apparentes du système de codage?

Code ASCII

- ASCII standard = codage sur 7 bits (0 à 127): les valeurs 0 à 31 sont utilisées pour les caractères de contrôle, les valeurs suivantes représentent les chiffres et les lettres
- Le code ASCII étendu utilise 8 bits (0 à 255): les 128 dernières valeurs dépendent du système

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header(SOH)	034	"	067	C	100	d	133	å	166	ª	199	Ä	232	ƒ
002	Start Of Text (STX)	035	#	068	D	101	e	134	ä	167	º	200	ℒ	233	ú
003	End Of Text (ETX)	036	\$	069	E	102	f	135	ç	168	¿	201	℞	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	è	169	®	202	℥	235	Ü
005	Enquiry	038	&	071	G	104	h	137	é	170	¬	203	℟	236	Ý
006	Acknowledge (ACK)	039		072	H	105	i	138	è	171	½	204	℟	237	Ÿ
007	Bell	040	(073	I	106	j	139	ï	172	¼	205	=	238	ˉ
008	Backspace (BS)	041)	074	J	107	k	140	î	173	ı	206	†	239	˘
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	⌘	240	-
010	Line Feed (LF)	043	+	076	L	109	m	142	À	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Á	176	⋈	209	Ð	242	_
012	Form Feed (FF)	045	-	078	N	111	o	144	É	177	⋈	210	È	243	¼
013	Carriage Return (CR)	046	.	079	O	112	p	145	æ	178	⋈	211	É	244	†
014	Shift Out	047	/	080	P	113	q	146	Æ	179		212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ø	180	‡	213	ı	246	÷
016	Dataline Escape (DLE)	049	1	082	R	115	s	148	ö	181	Á	214	í	247	,
017	DC 1 (XON)	050	2	083	S	116	t	149	ò	182	À	215	î	248	°
018	DC 2	051	3	084	T	117	u	150	û	183	Á	216	ï	249	˙
019	DC 3 (XOFF)	052	4	085	U	118	v	151	ü	184	®	217	Ĵ	250	˚
020	DC 4	053	5	086	V	119	w	152	ÿ	185	‡	218	Œ	251	˛
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186		219	■	252	˜
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	¶	220	■	253	˚
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	‡	221	ı	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	ı	255	
025	End Of Medium	058	:	091	[124		157	∅	190	¥	223	■		
026	Substitute	059	;	092	\	125	}	158	×	191	γ	224	ó		
027	Escape (ESC)	060	<	093]	126	~	159	f	192	ℒ	225	ß		
028	File Separator	061	=	094	^	127 (DEL)	␣	160	á	193	⊥	226	ô		
029	Group Separator	062	>	095	_	128	ç	161	í	194	⊥	227	ö		
030	Record Separator	063	?	096	`	129	ü	162	ó	195	‡	228	ø		
031	Unit Separator	064	@	097	a	130	é	163	ú	196	-	229	ő		
032	SPACE(SP)	065	A	098	b	131	â	164	ñ	197	+	230	μ		

OS & UNIX

- Le système d'exploitation (OS) permet à l'utilisateur d'interagir avec la machine
 - Créer des répertoires, lancer des programmes, copier et déplacer des fichiers
 - MS-DOS, Windows, Linux, MAC OS X...
 - Pour les TPS vous utiliserez UNIX qui est le système dominant en science et ingénierie

Commandes UNIX utiles

- whoami = opérateur courant
- pwd = chemin du répertoire courant
- mkdir = créer un répertoire
- ls = liste l'ensemble des fichiers et répertoires du répertoire courant
- rm = effacer un fichier
- rmdir = rm - rf= effacer un répertoire
- cp = copier un fichier
- mv = déplacer un fichier
- nedit = lance l'éditeur de texte nedit
- ctrl-z = stopper un process
- bg =mettre en tâche de fond
- man = aide en ligne d'une commande

Langage de programmation

- Langage de bas niveau (assembleur)
 - proche du langage machine (langage utilisé par l'ordinateur)
 - dépend du type de processeur
 - nécessaire de connaître le fonctionnement du processeur et l'adressage de la mémoire
 - non portable

Langage de programmation

- Langage de 'haut' niveau:
 - C, Fortran, Pascal...
 - contiennent beaucoup d'instructions
 - portables (car compilés et interprétés par des programmes indépendants de l'architecture)
 - plus proche du langage mathématique
 - C sera utilisé en TP

Algorithme & programme

- l'algorithme est indépendant du langage de programmation
 - définit la logique et les étapes pour résoudre le problème
 - proche du langage naturel (français)
 - proche du langage mathématique (opérations arithmétiques)
- On commence par définir l'algorithme avant d'écrire le programme

Algorithme & programme

- Une fois l'algorithme définit, on peut commencer à écrire le programme
 - dans le langage C (itératif) l'ordre des opérations est important
 - l'ordinateur va exécuter les programme pas à pas en suivant l'ordre de opérations que vous lui indiqué -> l'ordinateur ne réfléchit pas!!!

Structure d'un programme

```
# directives de compilation
```

```
//déclaration des fonctions
```

```
<type> func_name (<type paramètre>,<idem>...);
```

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

```
{
```

```
    déclaration des variables;
```

```
    // appel à fonction, opérations sur les variables etc
```

```
    block d'opérations logiques;
```

```
    [return 0;]
```

```
}
```

```
//définition des fonctions
```

```
<type> func_name(<type> par_name,<type> par_name...)
```

```
{
```

```
    déclarations et instructions;
```

```
    return X;
```

```
}
```

Structure d'un programme C

- Quel que soit le programme, il doit toujours suivre cette structure
 - si la structure n'est pas correcte, l'information manque au compilateur pour compiler le programme
- La syntaxe est également très importante
 - aucune erreur ne peut être tolérée: problème de compilation ou erreur de calcul

Structure d'un programme

- les bons programmes sont structurés en modules et fonction
 - plus lisible
 - plus facile à corriger en cas d'erreur
- Chaque module/fonction à un algorithme qui lui est propre
 - avec un nom et les paramètres d'E/S
 - l'algorithme général doit donc décomposer la tâche globale en sous-unités 'indépendantes'

Structure d'un programme

- Pourquoi décomposer?
 - meilleure lisibilité=meilleure compréhension de schéma global
 - conception plus aisée: démarche par étape, par tâche macroscopique qui est une démarche naturelle
 - mise au point des modules d'une façon indépendantes les uns des autres
 - maintenance et modification plus aisée
 - re-utilisabilité des sous-programmes (donc coût de production réduit)
 - abstraction: donner des noms à des actions complexes

Structures d'un programme

- Limites
 - un sous-programme doit effectuer une tâche précise sous peine d'augmenter la complexité du problème:
 - 1 sous-programme = 1 action complexe
 - 1 action complexe peut faire appel à d'autres sous-programme qui font d'autres actions complexes

Exercice algorithme

- Ex 3: écrivez l'algorithme qui permet de calculer les racines d'un polynôme de degré 2.
- Ex 4: écrire l'algorithme qui permet de comparer deux nombres et retourne le plus grand.