Programmation langage C
Section 10 : Consolidation des bases

Présentation de **Kevin TRANCHO**

dispensé en classe de seconde année

à l'**ESGI** Paris
(Année scolaire 2022 - 2023)

ESGI
école supérieure de
génie informatique

Présentation

Cours sur la programmation en langage C :

Objectifs :

Semestre 1 (bases de programmation) :

- Variables, conditions, boucles.

Semestre 2 (bases du langage C) :

- Fonctions, tableaux, pointeurs.

Semestre 3 (notions avancées du langage C) :

- Fichiers, structures, programmation modulaire, opérations bit-à-bit, types génériques.

Support de cours :

Support de cours :

Chaque section est distribuée sur MyGES pas à pas :

- Cours rédigé.
- Activités (non notées).
- Exercices (notés) : à rendre sur MyGES.
- Résumé du cours.
- Reprend les cours de l'an dernier et donne un rapide aperçu de SDL.

Support de cours :

Chaque section est distribuée sur MyGES pas à pas :

- Cours rédigé.
- Activités (non notées).
- Exercices (notés) : à rendre sur MyGES.
- Résumé du cours.
- Reprend les cours de l'an dernier et donne un rapide aperçu de SDL.

Support de cours :

Chaque section est distribuée sur MyGES pas à pas :

- Cours rédigé.
- Activités (non notées).
- Exercices (notés) : à rendre sur MyGES.
- Résumé du cours.
- Reprend les cours de l'an dernier et donne un rapide aperçu de SDL.

Support de cours :

Chaque section est distribuée sur MyGES pas à pas :

- Cours rédigé.
- Activités (non notées).
- Exercices (notés) : à rendre sur MyGES.
- Résumé du cours.
- Reprend les cours de l'an dernier et donne un rapide aperçu de SDL.

Support de cours :

Chaque section est distribuée sur MyGES pas à pas :

- Cours rédigé.
- Activités (non notées).
- Exercices (notés) : à rendre sur MyGES.
- Résumé du cours.
- Reprend les cours de l'an dernier et donne un rapide aperçu de SDL.

Support de cours :

Chaque section est distribuée sur MyGES pas à pas :

- Cours rédigé.
- Activités (non notées).
- Exercices (notés) : à rendre sur MyGES.
- Résumé du cours.
- Reprend les cours de l'an dernier et donne un rapide aperçu de SDL.

Évaluations Semestre 3 :

Déroulement des cours :

Avancement d'une notion :

- Présentation avec slides (comme celles-cis) avec étude et compilation de codes $\star^{\text{numéro}}$.
- Si exercices d'application pendant la présentation : temps donné puis correction.
- Temps d'autonomie pour travail sur exercices d'entraînement (notés et à rendre sur MyGES).
- Si terminé en avance : suite du support de cours ou projet C.
- Correction exercices après la deadline et début d'une nouvelle notion.

Quelques règles pour une bonne collaboration ensemble :

- Politesse.
- Respect.
- Rigueur.
- Authenticité.

Quand t'ouvres la porte en cours alors que t'es en retard

Quelques règles pour une bonne collaboration ensemble :

- Politesse.
- Respect.
- Rigueur.
- Authenticité.

Quelques règles pour une bonne collaboration ensemble :

- Politesse.
- Respect.
- Rigueur.
- Authenticité.

Quelques règles pour une bonne collaboration ensemble :

- Politesse.
- Respect.
- Rigueur.
- Authenticité.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

Coopération pour les rendus :

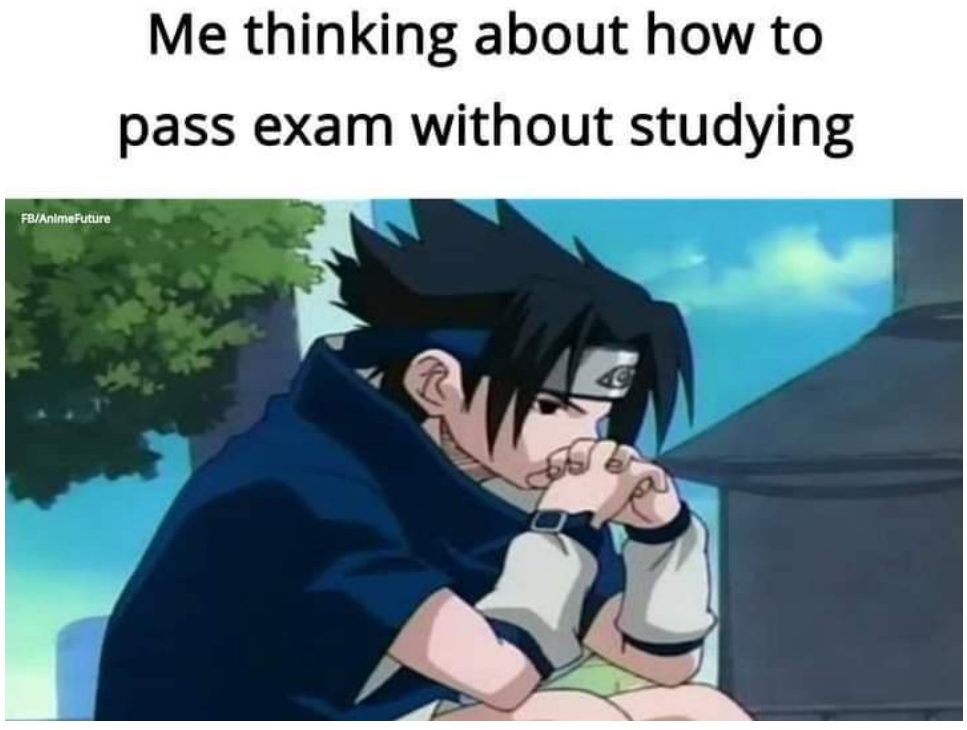
$$\frac{NoteRenduFinale}{20} = \max\left(\frac{\frac{NoteRendu}{20} \times \left\lfloor 22 - 2^{NombreParticipants} \right\rfloor}{20}, 0\right)$$

Nombre de participants	Note maximale
1	20
2	18
3	14
4	6
5+	0

- Possibilité de s'allier dans un groupe.
- Rendu autre que ce qui est attendu : pénalité.
- Rendus communs détectés additionnent leurs participants (>6 : note = 0).
- Rendre sa propre correction à l'enseignant : note = 0.

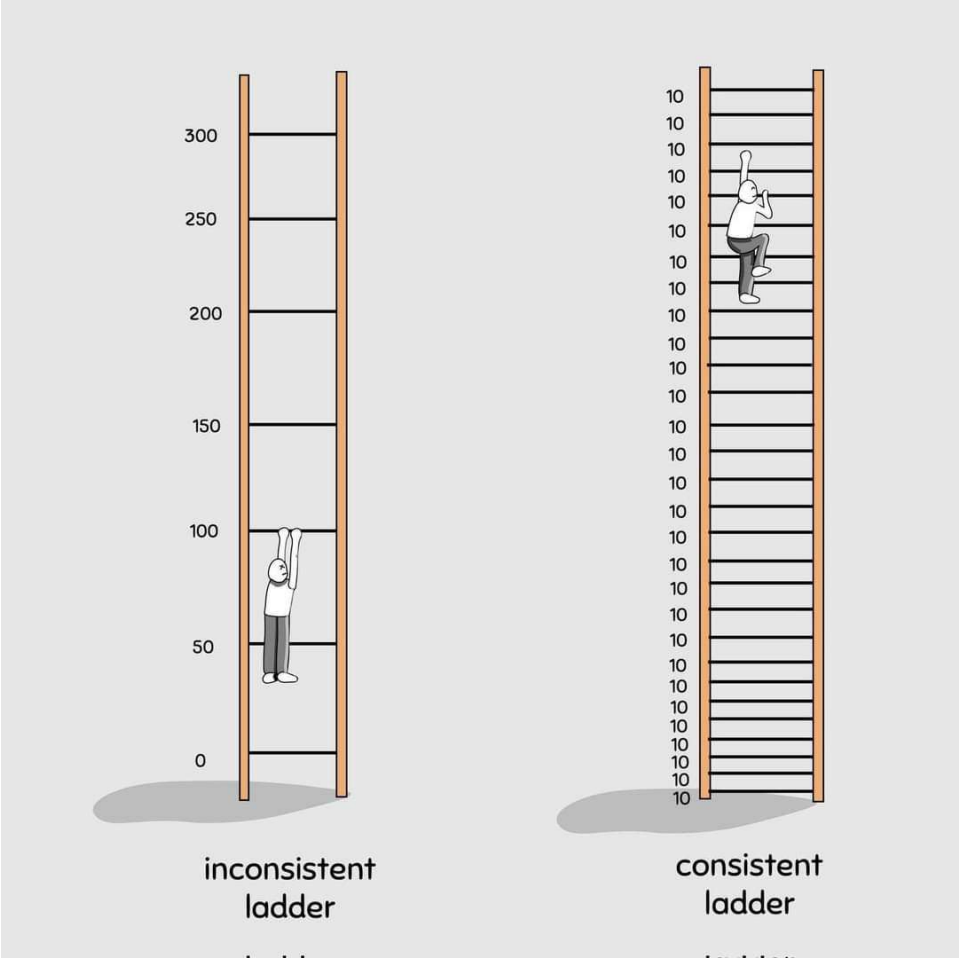
Conseils pour réussir son UE Langage C :

- Assiduité.
- Pratique.



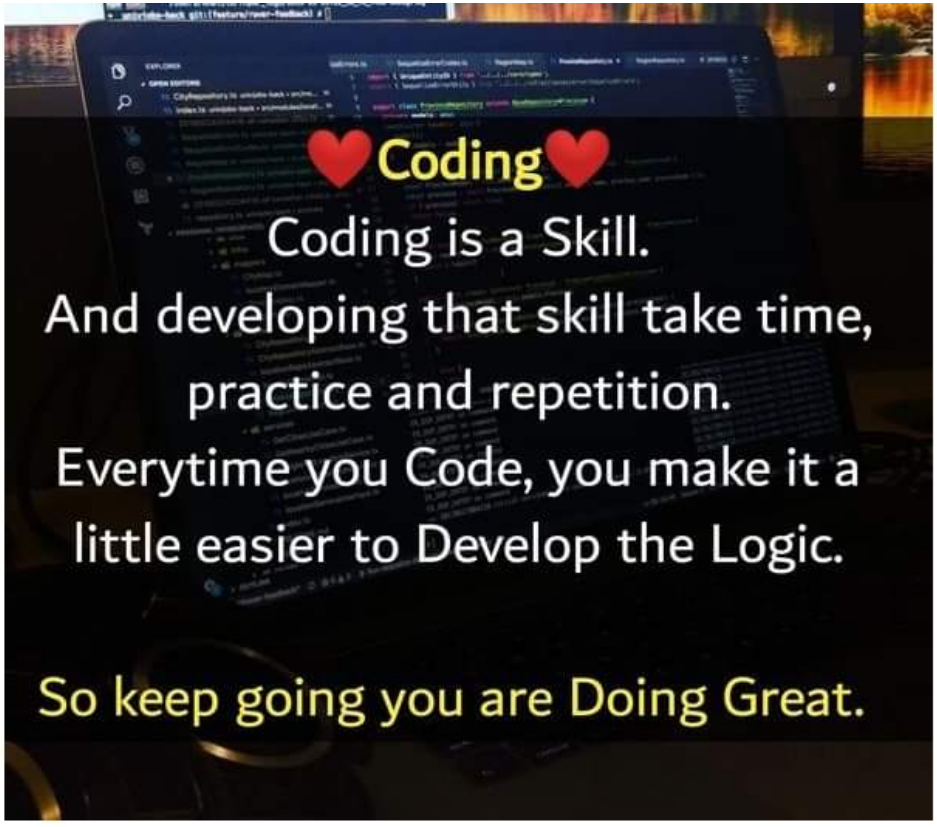
Conseils pour réussir son UE Langage C :

- Assiduité.
- Pratique.



Conseils pour réussir son UE Langage C :

- Assiduité.
- Pratique.



Projet : étapes

- **(Deadline : 03 Octobre 2022)** S'armer de deux camarades. (Non affecté sur MyGES ? Le professeur procédera à un matchmaking automatique).
- **(Deadline : 31 Octobre 2022)** Choisir une thématique et un sujet non commun puis les faire valider par le professeur. (Si non fait dans les temps, le sujet sera imposé).
- **(Deadline : 30 Novembre 2023)** Rendu d'une roadmap du projet :
  - répartition des tâches dans le groupe.
  - listing des fonctionnalités.
  - planification dans le temps.
- **(Deadline : 01 Février 2023)** Rendu d'un prototype du projet.
- **(Deadline : 20 Février 2023)** Rendu final du projet.
- **(24 Février 2023)** Soutenance.

Projet : étapes

- **(Deadline : 03 Octobre 2022)** S'armer de deux camarades.
  (Non affecté sur MyGES ? Le professeur procédera à un matchmaking automatique).
- **(Deadline : 31 Octobre 2022)** Choisir une thématique et un sujet non commun puis les faire valider par le professeur.
  (Si non fait dans les temps, le sujet sera imposé).
- **(Deadline : 30 Novembre 2023)** Rendu d'une roadmap du projet :
  - répartition des tâches dans le groupe.
  - listing des fonctionnalités.
  - planification dans le temps.
- **(Deadline : 01 Février 2023)** Rendu d'un prototype du projet.
- **(Deadline : 20 Février 2023)** Rendu final du projet.
- **(24 Février 2023)** Soutenance.

Projet : étapes

- **(Deadline : 03 Octobre 2022)** S'armer de deux camarades.
  (Non affecté sur MyGES ? Le professeur procédera à un matchmaking automatique).
- **(Deadline : 31 Octobre 2022)** Choisir une thématique et un sujet non commun puis les faire valider par le professeur.
  (Si non fait dans les temps, le sujet sera imposé).
- **(Deadline : 30 Novembre 2023)** Rendu d'une roadmap du projet :
  - répartition des tâches dans le groupe.
  - listing des fonctionnalités.
  - planification dans le temps.
- **(Deadline : 01 Février 2023)** Rendu d'un prototype du projet.
- **(Deadline : 20 Février 2023)** Rendu final du projet.
- **(24 Février 2023)** Soutenance.

Projet : étapes

- **(Deadline : 03 Octobre 2022)** S'armer de deux camarades.
  (Non affecté sur MyGES ? Le professeur procédera à un matchmaking automatique).
- **(Deadline : 31 Octobre 2022)** Choisir une thématique et un sujet non commun puis les faire valider par le professeur.
  (Si non fait dans les temps, le sujet sera imposé).
- **(Deadline : 30 Novembre 2023)** Rendu d'une roadmap du projet :
  - répartition des tâches dans le groupe.
  - listing des fonctionnalités.
  - planification dans le temps.
- **(Deadline : 01 Février 2023)** Rendu d'un prototype du projet.
- **(Deadline : 20 Février 2023)** Rendu final du projet.
- **(24 Février 2023)** Soutenance.

Projet : étapes

- **(Deadline : 03 Octobre 2022)** S'armer de deux camarades.
  (Non affecté sur MyGES ? Le professeur procédera à un matchmaking automatique).
- **(Deadline : 31 Octobre 2022)** Choisir une thématique et un sujet non commun puis les faire valider par le professeur.
  (Si non fait dans les temps, le sujet sera imposé).
- **(Deadline : 30 Novembre 2023)** Rendu d'une roadmap du projet :
  - répartition des tâches dans le groupe.
  - listing des fonctionnalités.
  - planification dans le temps.
- **(Deadline : 01 Février 2023)** Rendu d'un prototype du projet.
- **(Deadline : 20 Février 2023)** Rendu final du projet.
- **(24 Février 2023)** Soutenance.

Projet : étapes

- **(Deadline : 03 Octobre 2022)** S'armer de deux camarades.
  (Non affecté sur MyGES ? Le professeur procédera à un matchmaking automatique).
- **(Deadline : 31 Octobre 2022)** Choisir une thématique et un sujet non commun puis les faire valider par le professeur.
  (Si non fait dans les temps, le sujet sera imposé).
- **(Deadline : 30 Novembre 2023)** Rendu d'une roadmap du projet :
  - répartition des tâches dans le groupe.
  - listing des fonctionnalités.
  - planification dans le temps.
- **(Deadline : 01 Février 2023)** Rendu d'un prototype du projet.
- **(Deadline : 20 Février 2023)** Rendu final du projet.
- **(24 Février 2023)** Soutenance.

Projet : évaluation

10 points de groupe **11 points individuels**

- **(... / 9 points) Code :**
  7 points de groupe 2 points individuels

  (Qualité code, documentation, découpe en module, structures de données, généricité, utilisation de bibliothèques et autres)
- **(... / 7 points) Fonctionnalités :**
  1 points de groupe 6 points individuels

  (Efficacité, fiabilité, atteinte des objectifs du sujet / roadmap, qualité, niveau technique et autres)
- **(... / 5 points) Soutenance et rapport :**
  2 points de groupe 3 points individuels

  (Organisation du code, de l'équipe, investissement personnel)

Projet : évaluation

10 points de groupe **11 points individuels**

- **(... / 9 points) Code** :
  7 points de groupe 2 points individuels

  (Qualité code, documentation, découpe en module, structures de données, généricité, utilisation de bibliothèques et autres)
- **(... / 7 points) Fonctionnalités** :
  1 points de groupe 6 points individuels

  (Efficacité, fiabilité, atteinte des objectifs du sujet / roadmap, qualité, niveau technique et autres)
- **(... / 5 points) Soutenance et rapport** :
  2 points de groupe 3 points individuels

  (Organisation du code, de l'équipe, investissement personnel)

Projet : évaluation

10 points de groupe **11 points individuels**

- **(... / 9 points) Code** :
  7 points de groupe 2 points individuels

  (Qualité code, documentation, découpe en module, structures de données, généricité, utilisation de bibliothèques et autres)

- **(... / 7 points) Fonctionnalités** :
  1 points de groupe 6 points individuels

  (Efficacité, fiabilité, atteinte des objectifs du sujet / roadmap, qualité, niveau technique et autres)

- **(... / 5 points) Soutenance et rapport** :
  2 points de groupe 3 points individuels

  (Organisation du code, de l'équipe, investissement personnel)

Projet : évaluation

10 points de groupe **11 points individuels**

- **(... / 9 points) Code** :
  7 points de groupe 2 points individuels

  (Qualité code, documentation, découpe en module, structures de données, généricité, utilisation de bibliothèques et autres)

- **(... / 7 points) Fonctionnalités** :
  1 points de groupe 6 points individuels

  (Efficacité, fiabilité, atteinte des objectifs du sujet / roadmap, qualité, niveau technique et autres)

- **(... / 5 points) Soutenance et rapport** :
  2 points de groupe 3 points individuels

  (Organisation du code, de l'équipe, investissement personnel)

Consolidation des bases

Prêt à embrayer sur du langage C ?

On reprend rapidement les concepts vu en première année (Pour plus de détails reprendre les deux premières parties du support de cours).

Consolidation des bases

Prêt à embrayer sur du langage C ?

On reprend rapidement les concepts vu en première année (Pour plus de détails reprendre les deux premières parties du support de cours).

Code source en langage C

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

/* Déclaration des fonctions */

int main() {

  /* Instructions de l'application */

  exit(EXIT_SUCCESS);
}
```

Compilation code source en langage C

```
gcc -o monProgramme main.c

./monProgramme
```

Si non installés, avoir un compilateur `gcc` ou `clang` sous Linux, Mac ou Windows (via MSYS2) : voir support de cours.

Variables : types atomiques

Entiers :

```
char  /* 1 octet */
short /* 2 octets */
int   /* 4 octets */
long  /* 8 octets */
unsigned type
      /* positif */
```

Flottants :

```
float       /*  4 octets */
double      /*  8 octets */
long double /* 16 octets */
```

Afficher des valeurs dans la console via printf

```c
printf("caractere : \'%c\'\n", '@');
printf("entier : %d\n", 42);
printf("hexadécimal : %x\n", 0x2a);
printf("entier positif : %u\n", 3000000000u);
printf("entier long : %ld\n", 42000000000);
printf("flottant : %g\n", 3.14);
printf("adresse : %p\n", NULL);
```

Lire des valeurs depuis la console via scanf

```
int entier;
scanf("%d", &entier);
float flottant;
scanf("%f", &flottant);
char caractere;
scanf(" %c", &caractere);
```

Opérations de base sur types atomiques

```
/*first et second deux variables de type entier ou
↪  flottant*/
first + second /* addition */
first - second /* soustraction */
first * second /* multiplication */
first / second /* division */
first % second /* modulo : entiers */
```

Coercition : nécessité de palier la limitation d'un type

```c
int first, second;
/* ... */
long multiplication = (long)first * second;
/* dépassement de capacité d'un entier */
float division = (float)first / second;
/* division entière réalisée */
```

Réaffectation par opérateur (incrémentation)

```c
int valeur = 1;
valeur = valeur + 1; /* ajoute 1 */
valeur += 1;         /* ajoute 1 */
valeur++;            /* ajoute 1 */
++valeur;            /* ajoute 1 */
```

if : structure de contrôle pour disjonction $\star^1$

```c
int first, second;
scanf("%d %d", &first, &second);
if(first > second) {
  printf("%d est plus grand.\n", first);
} else if(first < second) {
  printf("%d est plus grand.\n", second);
} else {
  printf("les deux sont égaux.\n");
}
```

Opérateurs de comparaison

```
/* égalité : */
a == b
/* plus petit strict : */
a < b
/* plus petit ou égal : */
a <= b
```

```
/* différence : */
a != b
/* plus grand strict : */
a > b
/* plus grand ou égal : */
a >= b
```

Opérateurs booléens

```
a && b /* vrai lorsque les deux le sont */

a || b /* vrai lorsque d'un l'est */

! a /* vrai lorsque faux et réciproquement */
```

Opérateur ternaire : remplacement d'un if

```
if(a < b) {
  res = a;
} else {
  res = b;
}
```

$\Leftrightarrow$

```
res = (a < b) ? a : b;
```

Structure de contrôle par branchements : switch

```
switch(expression) {
        case 1 : {
                /* instructions */
        } break;
        case 2 : {
                /* instructions */
        } break;
        default : {
                /* instructions */
        }
}
```

Structures de répétition

- `while` pour répéter les instructions tant qu'une condition est vraie.

```
while(condition) {
        /* instructions */
}
```

- `do-while` pour répéter à nouveau les instructions sous condition.

```
do {
        /* instructions */
} while(condition);
```

- `for` lorsque la boucle while peut s'écrire avec une initialisation et une évolution des données utilisées pour la condition.

```
for(initialisation; condition; evolution) {
        /* instructions */
}
```

Structures de répétition

- `while` pour répéter les instructions tant qu'une condition est vraie.

```
while(condition) {
        /* instructions */
}
```

- `do-while` pour répéter à nouveau les instructions sous condition.

```
do {
        /* instructions */
} while(condition);
```

- `for` lorsque la boucle while peut s'écrire avec une initialisation et une évolution des données utilisées pour la condition.

```
for(initialisation; condition; evolution) {
        /* instructions */
}
```

Structures de répétition

- `while` pour répéter les instructions tant qu'une condition est vraie.

```
while(condition) {
        /* instructions */
}
```

- `do-while` pour répéter à nouveau les instructions sous condition.

```
do {
        /* instructions */
} while(condition);
```

- `for` lorsque la boucle while peut s'écrire avec une initialisation et une évolution des données utilisées pour la condition.

```
for(initialisation; condition; evolution) {
        /* instructions */
}
```

Définition d'une fonction

```
typeRetour nomFonction(/* paramètres */) {
        /* instructions */
}

/* Exemple de fonction d'addition d'entiers : */
int addition(int first, int second) {
        int res; /* variable locale à la fonction */
        res = first + second;
        return res; /* renvoi lors de l'appel */
}
```

Déclaration d'une fonction

```c
/* Exemple de fonction d'addition d'entiers : */
/* déclaration */
int addition(int, int);
int main() {
        /* appel */
        int deux = addition(1, 1);
        exit(EXIT_SUCCESS);
}
/* définition */
int addition(int first, int second) {
        return first + second;
}
```

Tableaux

```
type tableau[TAILLE_CONSTANTE];
type tableau[] = {valeur1, valeur2, ..., valeurN};

int liste[] = {1, 2, 3};
liste[1]; /* accès au second élément : d'indice 1 */
```

Tableaux : chaînes de caractères

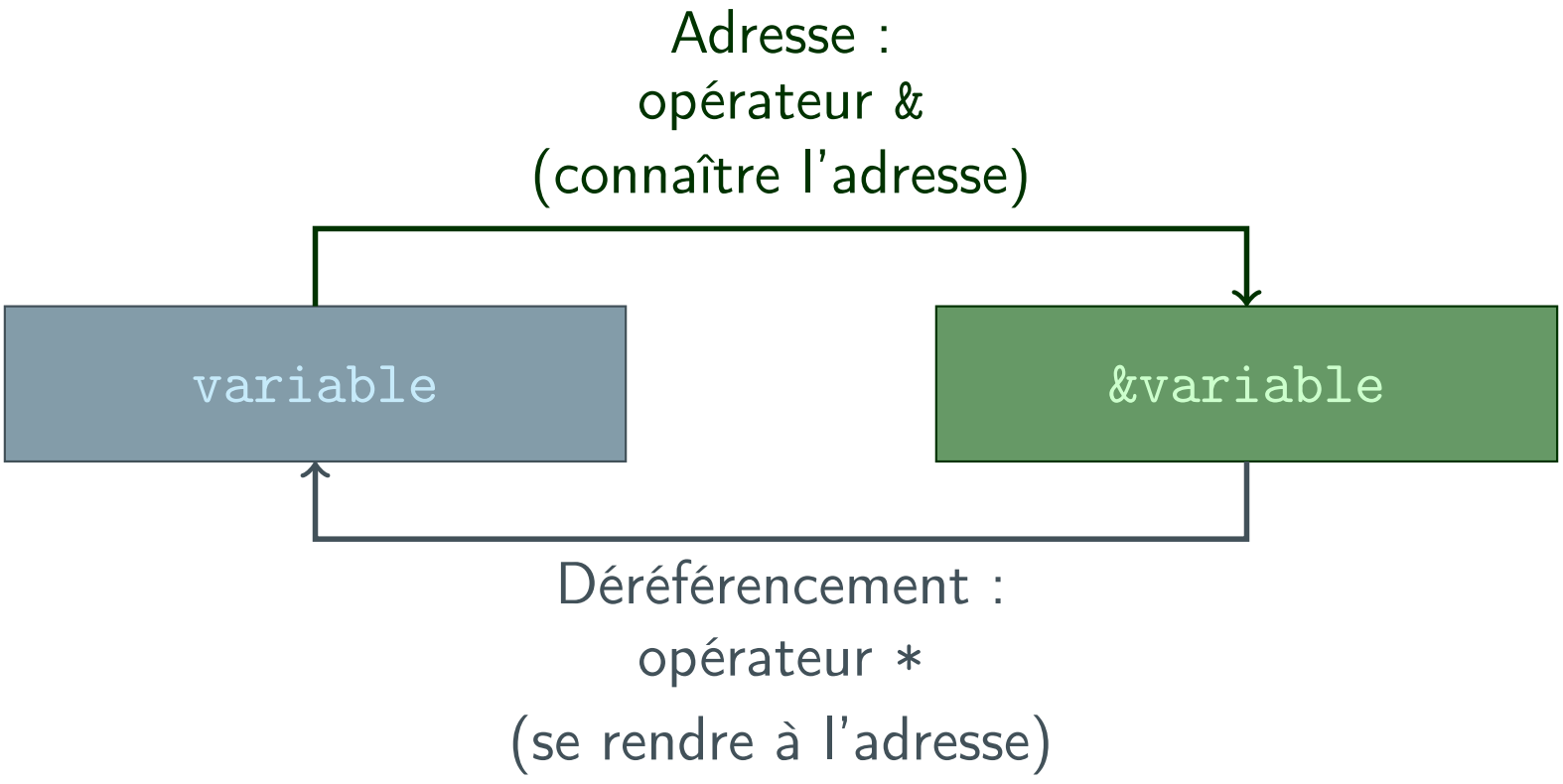
Les tableaux sont utilisés pour gérer les chaînes de caractères (se terminant par un marque de fin `'\0'`).

```
char chaine[] = "Hello ESGI !";

/* affichage de chaine : */
printf("%s\n", chaine);

/* lecture d'un mot au clavier : */
scanf("%s", chaine);
```

Tableaux à plusieurs dimensions

```
/* tableau à deux dimensions */
int grille[HAUTEUR][LARGEUR] = {
  {0, 0, 0, 0},
  {0, 1, 2, 0},
  {0, 0, 0, 0}
};
grille[ligne][colonne]; /* accès au tableau */
/* passage d'un tableau à deux dimensions */
void afficherGrille(int largeur, int hauteur,
    grille[hauteur][largeur]) {
  /* instructions */
}
```

Adresse d'une variable



Pointeur sur une adresse $\star^2$

```c
int variable = 42;
int * pointeur = &variable;
*pointeur = 1337;
printf("%d\n", variable); /* affiche 1337 */
```

Pointeur sur un tableau

```c
int tableau[] = {1, 2, 3, -1};
int * pointeur = tableau;
int i;
for(i = 0; pointeur[i] >= 0; ++i) {
  printf("%d\n", pointeur[i]);
}
```

Arithmétique des pointeurs $\star^3$

```
char texte[] = "Hello !";
char * pointeur = NULL;
for(pointeur = texte; *pointeur != '\0'; ++pointeur)
 ↪  {
  if(*pointeur >= 'a' && *pointeur <= 'z')
    *pointeur += 'A' - 'a';
}
printf("%s\n", texte); /* affiche "HELLO !" */
```

Allocation dynamique $\star^4$

```c
float * notes = NULL;
float somme = 0;
int nombre;
int i;
printf("Combien de CC ? ");
scanf("%d", &nombre);
if(nombre <= 0) {
  printf("Pas de notes pas de moyenne.\n");
  exit(EXIT_FAILURE);
}
/* allocation dynamique depuis le nombre donné par
↪  l'utilisateur */
if((notes = (float *)malloc(sizeof(float) * nombre)) == NULL) {
  printf("Erreur d'allocation.\n");
  exit(EXIT_FAILURE);
}
```

Allocation dynamique

```c
/* ... */
for(i = 0; i < nombre; ++i) {
  scanf("%f", notes + i);
  somme += notes[i];
}
printf("La moyenne de ");
for(i = 0; i < nombre; ++i) {
  if(i && i == nombre - 1) printf(" et ");
  else if(i > 0) printf(", ");
  printf("%g", notes[i]);
}
printf(" est %g\n", somme / nombre);

free(notes);
notes = NULL;
exit(EXIT_SUCCESS);
```

Allocation dynamique

```
/* allouer une plage mémoire */
malloc(/*taille mémoire en octets*/)

/* allouer un tableau avec chaque élément à 0 */
calloc(/*taille tableau*/, /*taille élément*/)

/* modifier la taille d'une plage allouée */
realloc(/*plage à modifier*/, /*nouvelle taille en
↪  octets*/)
```

Questions

Avez-vous des questions ?

Exercices

- Travailler sur les exercices de consolidation des bases (section 10) du support de cours.
- Si les exercices de la section 10 sont terminés :
  - Avancer sur les sections 11 et 12.
  - Si cours terminé : Avancer sur le projet.
  - Si projet terminé avec certitude de 21 / 20 : le pousser plus loin.

Annexe

When the code is a mess
but it's working anyway