Programmer en langage C: un outil pour physicien - Introduction

Laurent de FORGES de PARNY

Cours Licence 3 Physique Nice 2010-2011

email: de.forges.de.parny.laurent@etu.unice.fr
Université de Nice Sophia-Antipolis

Institut Non Linéaire de Nice

Labo mixte CNRS + UNS

- Environ 30 chercheurs + 10 étudiants en thèse

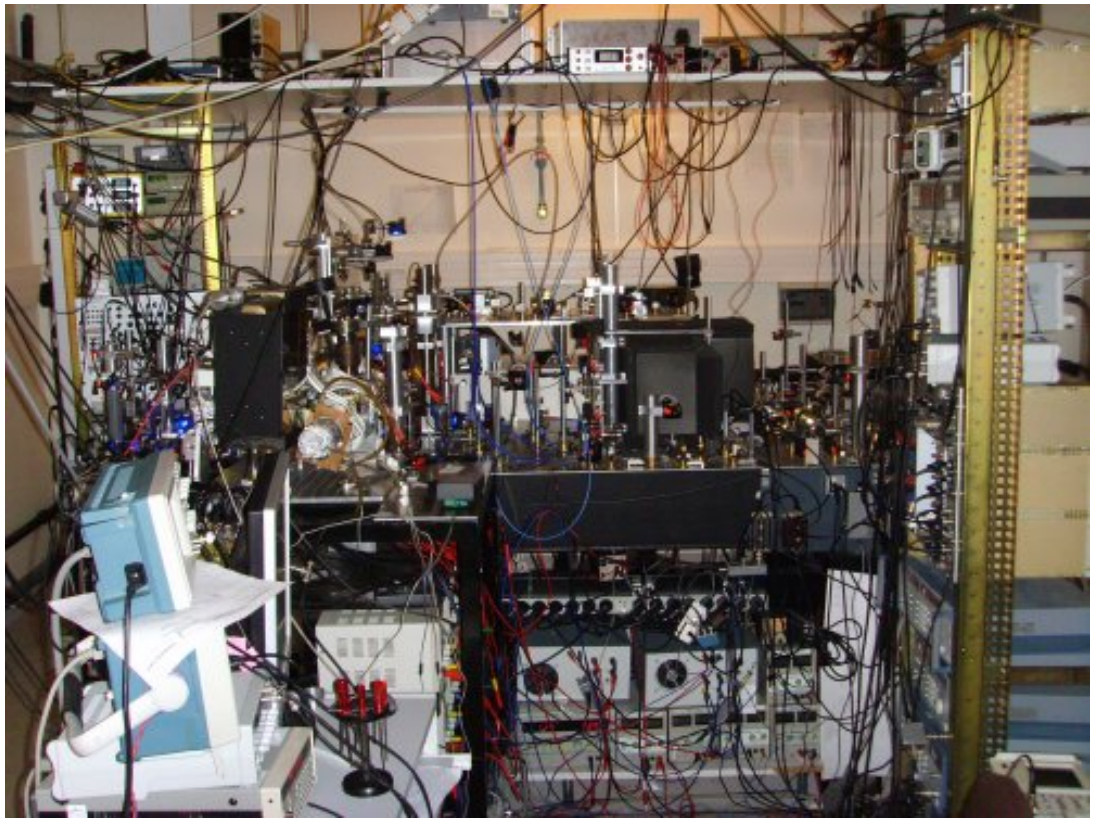
Deux principaux domaines de recherches (en gros)

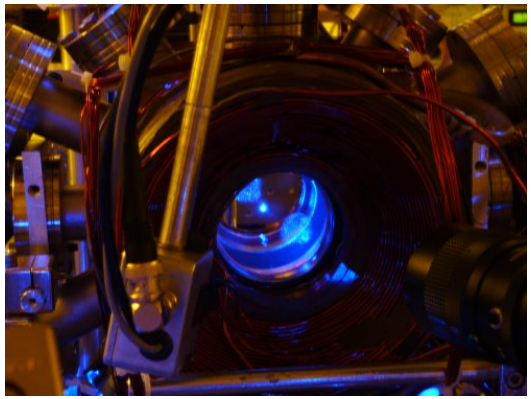
- Physique non linéaire (Optique, Hydrodynamique, BioPhysique, ...)
- Physique quantique (Atomes froids, Systèmes fortement corrélés, ...)

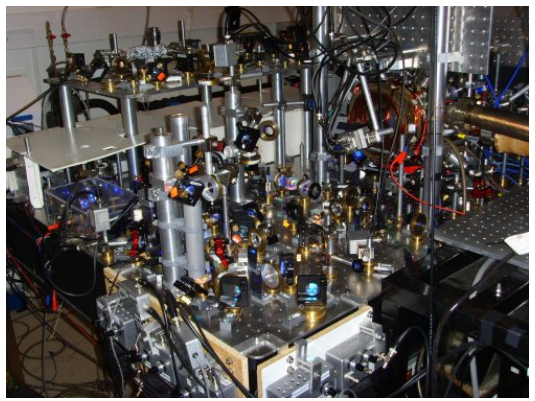
$\longrightarrow$ Expériences, simulations et théorie

Ma thèse: Etude des phases quantiques exotiques dans les condensats de Bose-Einstein à spin-1

On refroidit des atomes à $T \sim 1\ \mu K$

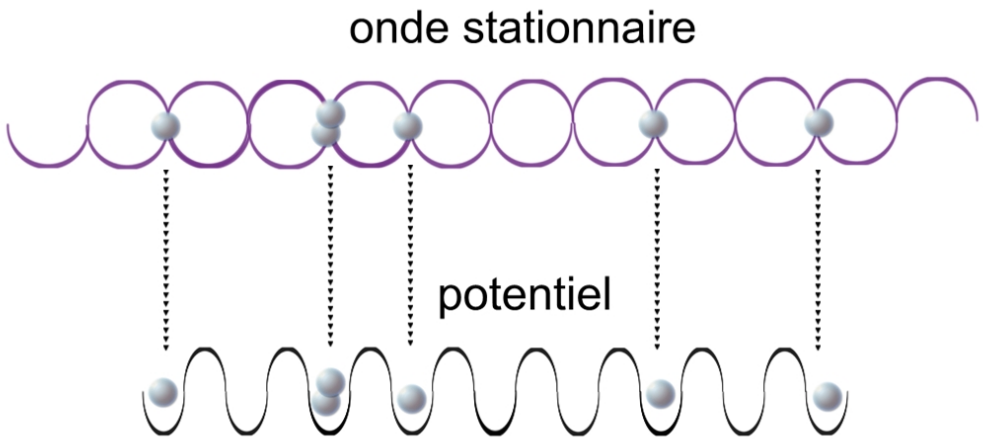






Réseau optique : potentiel sinusoïdal

→ Particules piégées sur les noeuds du réseau (minimum de l'énergie) :



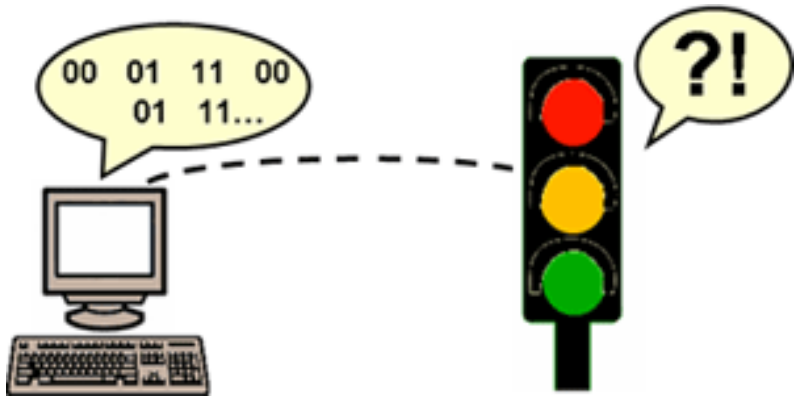
→ Etude numérique du système (étude théorique, permet de motiver ou non une manip ...)

Le numérique c'est quoi ?

- Numérique : langage composé de chiffres, code constitué de 0 et de 1

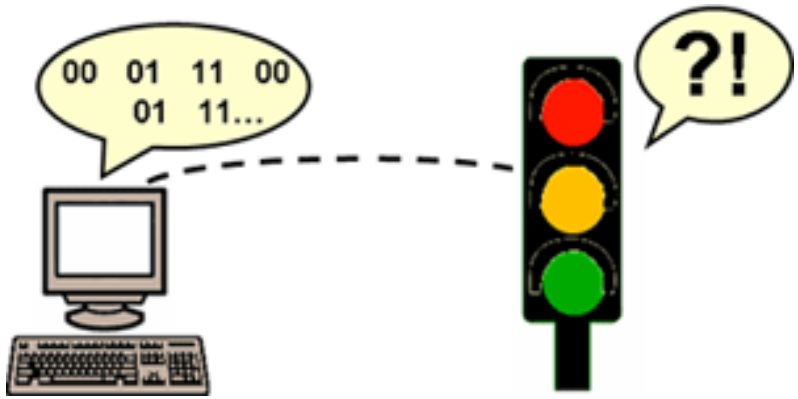
Le numérique c'est quoi ?

- Numérique : langage composé de chiffres, code constitué de 0 et de 1
- Exemple :
  feu numérique pour la circulation, commandé par un ordinateur
  rouge = 00, orange = 01, vert =11



Le numérique c'est quoi ?

- Numérique : langage composé de chiffres, code constitué de 0 et de 1
- Exemple :
  feu numérique pour la circulation, commandé par un ordinateur
  rouge = 00, orange = 01, vert =11



- Il faut apprendre à l'ordinateur et au feu à parler ce code : programme informatique (logiciel)

Le numérique c'est quoi ?

- Machines numériques : appareil photo, camescope, écran plat, ordinateur ...
  Exemple : 000000000100101100010101 = vert foncé

Le numérique c'est quoi ?

- Machines numériques : appareil photo, camescope, écran plat, ordinateur ...
  Exemple : 00000000010010110001010 1 = vert foncé

- Analogique (signal continu) : contraire du numérique (signal échantillonné)
  Exemples analogiques : microphone, montre mécanique, ampèremètre, platine cassette ...

Le numérique c'est quoi ?

- Machines numériques : appareil photo, camescope, écran plat, ordinateur ...
  Exemple : 000000000100101100010101 = vert foncé
- Analogique (signal continu) : contraire du numérique (signal échantillonné)
  Exemples analogiques : microphone, montre mécanique, ampèremètre, platine cassette ...
- Numérique : moins réaliste mais plus pratique !

Le numérique autour de nous ?

- Ordinateur, appareil photo, camescope, tel. portable, wifi, internet, TV, horloge ...

Le numérique autour de nous ?

- Ordinateur, appareil photo, camescope, tel. portable, wifi, internet, TV, horloge ...
- Voiture, moto, avion, train, fusée, GPS (Global Positioning System) ...

Le numérique autour de nous ?

- Ordinateur, appareil photo, camescope, tel. portable, wifi, internet, TV, horloge ...
- Voiture, moto, avion, train, fusée, GPS (Global Positioning System) ...
- Trafic routier, cinéma, météo, banque, sécurité civile, armenent, médecine ...

$\longrightarrow$ **Numeric is everywhere**

Le monde sans numérique ?

Le monde sans numérique ?

Essais nucléaires français

- Désert algérien : 1960 $\longrightarrow$ 1966

  $\longrightarrow$ 1ere bombe atomique : "Gerboise" en 1960

Essais nucléaires français

- Désert algérien : 1960 $\longrightarrow$ 1966

  $\longrightarrow$ 1ere bombe atomique : "Gerboise" en 1960

  $\longrightarrow$ 4 essais aériens + 13 souterrains

Essais nucléaires français

- Désert algérien : 1960 $\longrightarrow$ 1966

  $\longrightarrow$ 1ere bombe atomique : "Gerboise" en 1960

  $\longrightarrow$ 4 essais aériens + 13 souterrains

  $\longrightarrow$ 1 catastrophe : fuite du nuage nucléaire

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga

  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga
  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols
  $\longrightarrow$ 1 catastrophe : dispersion radioactive (cyclone à Mururoa 1981)

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga
  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols
  $\longrightarrow$ 1 catastrophe : dispersion radioactive (cyclone à Mururoa 1981)
  $\longrightarrow$ 1968 : 1ere bombe H (thermonuclaire) française à Fangatanga

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga
  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols
  $\longrightarrow$ 1 catastrophe : dispersion radioactive (cyclone à Mururoa 1981)
  $\longrightarrow$ 1968 : 1ere bombe H (thermonuclaire) française à Fangatanga
  $\longrightarrow$ 1992 : utiliser la simulation numérique ?

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga
  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols
  $\longrightarrow$ 1 catastrophe : dispersion radioactive (cyclone à Mururoa 1981)
  $\longrightarrow$ 1968 : 1ere bombe H (thermonuclaire) française à Fangatanga
  $\longrightarrow$ 1992 : utiliser la simulation numérique ?
  $\longrightarrow$ 1992 à 1996 : 7 derniers essais nucléaires (Chirac) pour obtenir des données physiques

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga
  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols
  $\longrightarrow$ 1 catastrophe : dispersion radioactive (cyclone à Mururoa 1981)
  $\longrightarrow$ 1968 : 1ere bombe H (thermonuclaire) française à Fangatanga
  $\longrightarrow$ 1992 : utiliser la simulation numérique ?
  $\longrightarrow$ 1992 à 1996 : 7 derniers essais nucléaires (Chirac) pour obtenir des données physiques
  $\longrightarrow$ arrêt total le 26 sept. 1996 pour la France

Essais nucléaires français

- Polynésie française : 1966 $\longrightarrow$ 1996

  $\longrightarrow$ 193 tirs à Mururoa et Fangatanga
  $\longrightarrow$ affaissement de la couronne corallienne & contamination des sols
  $\longrightarrow$ 1 catastrophe : dispersion radioactive (cyclone à Mururoa 1981)
  $\longrightarrow$ 1968 : 1ere bombe H (thermonuclaire) française à Fangatanga
  $\longrightarrow$ 1992 : utiliser la simulation numérique ?
  $\longrightarrow$ 1992 à 1996 : 7 derniers essais nucléaires (Chirac) pour obtenir des données physiques
  $\longrightarrow$ arrêt total le 26 sept. 1996 pour la France
  $\longrightarrow$ conséquences sanitaires : 76 000 personnes

Apogée de la folie : "Tsar Bomba"

- Nouvelle-Zemble, grand nord Russe : 31 oct. 1961

  $\longrightarrow$ Bombe H la plus puissante tirée : 57 Mégatonnes

Apogée de la folie : "Tsar Bomba"

- Nouvelle-Zemble, grand nord Russe : 31 oct. 1961

  $\longrightarrow$ Bombe H la plus puissante tirée : 57 Mégatonnes

  $\longrightarrow$ 2400 fois la bombe A d'Hiroshima

Apogée de la folie : "Tsar Bomba"

- Nouvelle-Zemble, grand nord Russe : 31 oct. 1961

  $\longrightarrow$ Bombe H la plus puissante tirée : 57 Mégatonnes

  $\longrightarrow$ 2400 fois la bombe A d'Hiroshima

  $\longrightarrow$ Onde de choc : 3 fois le tour de la Terre

Apogée de la folie : "Tsar Bomba"

- Nouvelle-Zemble, grand nord Russe : 31 oct. 1961

  $\longrightarrow$ Bombe H la plus puissante tirée : 57 Mégatonnes
  $\longrightarrow$ 2400 fois la bombe A d'Hiroshima
  $\longrightarrow$ Onde de choc : 3 fois le tour de la Terre
  $\longrightarrow$ Zone détruite : équivalente à Paris

  Video : Tsar Bomba

Programme de simulation nucléaire français

- Commisariat à l'Energie Nucléaire (CEA)

  $\longrightarrow$ simuler les étapes d'une explosion thermonucléaire

Programme de simulation nucléaire français

- Commisariat à l'Energie Nucléaire (CEA)

  $\longrightarrow$ simuler les étapes d'une explosion thermonucléaire

  $\longrightarrow$ validation des modèles via les données des campagnes du Pacifique

Programme de simulation nucléaire français

- Commisariat à l'Energie Nucléaire (CEA)

  $\longrightarrow$ simuler les étapes d'une explosion thermonucléaire
  $\longrightarrow$ validation des modèles via les données des campagnes du Pacifique

- 3 types de systèmes :

  $\longrightarrow$ AIRIX : Accélérateur à Induction de Radiographie pour l'Imagerie X
  $\longrightarrow$ Laser Mégajoule : 240 faisceaux lasers convergeant !
  $\longrightarrow$ TERA : logiciels de calculs extrêmement puissants, 5000 milliards d'opérations / seconde. 2560 processeurs dans 60 m^2

Modélisation et conception

- Mécanique : Radial Engine
- Animation 3D :
  - $\longrightarrow$ Superstructures (7 mins 20)
  - $\longrightarrow$ Lord of the Rings (2 mins 08)
  - $\longrightarrow$ Diplodocus
  - $\longrightarrow$ Kung Fu Panda (59 mins, 1H04, 1H18)

Modélisation et conception

- Mécanique : Radial Engine
- Animation 3D :
  $\longrightarrow$ Superstructures (7 mins 20)
  $\longrightarrow$ Lord of the Rings (2 mins 08)
  $\longrightarrow$ Diplodocus
  $\longrightarrow$ Kung Fu Panda (59 mins, 1H04, 1H18)

Animations soumises aux lois de la Physique (sauf Légolas !)
$\longrightarrow$ intégrer, résoudre des équations ...

Simulation numérique en L3 Physique

- 10h cours "rappel"

Simulation numérique en L3 Physique

- 10h cours "rappel"
- Blocs cours spécifiques + TP

Simulation numérique en L3 Physique

- 10h cours "rappel"
- Blocs cours spécifiques + TP
- Projet numérique en trinôme

Simulation numérique en L3 Physique

- 10h cours "rappel"
- Blocs cours spécifiques + TP
- Projet numérique en trinôme

But du projet :

- Etudier un problème de Physique difficilement soluble analytiquement
  $\longrightarrow$ onde, méca, optique, thermo, hydro, astro ...

Simulation numérique en L3 Physique

- 10h cours "rappel"
- Blocs cours spécifiques + TP
- Projet numérique en trinôme

But du projet :

- Etudier un problème de Physique difficilement soluble analytiquement
  $\longrightarrow$ onde, méca, optique, thermo, hydro, astro ...
- Exemples : sujets et programmes

Simulation numérique en L3 Physique

- 10h cours "rappel"
- Blocs cours spécifiques + TP
- Projet numérique en trinôme

But du projet :

- Etudier un problème de Physique difficilement soluble analytiquement
  $\longrightarrow$ onde, méca, optique, thermo, hydro, astro ...
- Exemples : sujets et programmes
- Présentation orale devant jury "très sévère"

But : "RAPPEL" des bases du langage C

- Qu'est-ce que le langage C ? Comment utiliser un programme ?
  $\longrightarrow$ un premier programme, comment utiliser un compilateur

But : "RAPPEL" des bases du langage C

- Qu'est-ce que le langage C ? Comment utiliser un programme ?
  $\longrightarrow$ un premier programme, comment utiliser un compilateur
- Concepts fondamentaux du langage C
  $\longrightarrow$ déclarations et types des variables

But : "RAPPEL" des bases du langage C

- Qu'est-ce que le langage C ? Comment utiliser un programme ?
  $\longrightarrow$ un premier programme, comment utiliser un compilateur
- Concepts fondamentaux du langage C
  $\longrightarrow$ déclarations et types des variables
- Lecture et écriture par un programme, manipuler des données
  $\longrightarrow$ entrées (scanf) et sorties (printf), opérations de base

But : "RAPPEL" des bases du langage C

- Qu'est-ce que le langage C ? Comment utiliser un programme ?
  $\longrightarrow$ un premier programme, comment utiliser un compilateur
- Concepts fondamentaux du langage C
  $\longrightarrow$ déclarations et types des variables
- Lecture et écriture par un programme, manipuler des données
  $\longrightarrow$ entrées (scanf) et sorties (printf), opérations de base
- Instructions et opérateurs logiques
  $\longrightarrow$ fonctions itératives (for, while, do-while) et instructions conditionnelles (if, if-else)

But : "RAPPEL" des bases du langage C

- Qu'est-ce que le langage C ? Comment utiliser un programme ?
  $\longrightarrow$ un premier programme, comment utiliser un compilateur
- Concepts fondamentaux du langage C
  $\longrightarrow$ déclarations et types des variables
- Lecture et écriture par un programme, manipuler des données
  $\longrightarrow$ entrées (scanf) et sorties (printf), opérations de base
- Instructions et opérateurs logiques
  $\longrightarrow$ fonctions itératives (for, while, do-while) et instructions conditionnelles (if, if-else)
- Stockage des données, fonctions C et fonctions mathématiques
  $\longrightarrow$ les tableaux, structurer un programme, fonctions sin, cos, sqrt ...

Le langage C ?

- Pourquoi le langage C ?

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant
  $\longrightarrow$ abordable car proche de l'anglais

Le langage C ?

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant
  $\longrightarrow$ abordable car proche de l'anglais
  $\longrightarrow$ compilable sur tous les systèmes UNIX (Mandriva, RedHat, Mac OS, Ubuntu, Debian, Mandrake ...)

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant
  $\longrightarrow$ abordable car proche de l'anglais
  $\longrightarrow$ compilable sur tous les systèmes UNIX (Mandriva, RedHat, Mac OS, Ubuntu, Debian, Mandrake ...)
  $\longrightarrow$ maîtrise du C : Perl, C++, Java ... facile à apprendre

Le langage C ?

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant
  $\longrightarrow$ abordable car proche de l'anglais
  $\longrightarrow$ compilable sur tous les systèmes UNIX (Mandriva, RedHat, Mac OS, Ubuntu, Debian, Mandrake ...)
  $\longrightarrow$ maîtrise du C : Perl, C++, Java ... facile à apprendre

- Mais ...

Le langage C ?

- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant
  $\longrightarrow$ abordable car proche de l'anglais
  $\longrightarrow$ compilable sur tous les systèmes UNIX (Mandriva, RedHat, Mac OS, Ubuntu, Debian, Mandrake ...)
  $\longrightarrow$ maîtrise du C : Perl, C++, Java ... facile à apprendre

- Mais ...

  $\longrightarrow$ langage exigeant et explicite (permet d'être rigoureux)

Le langage C ?

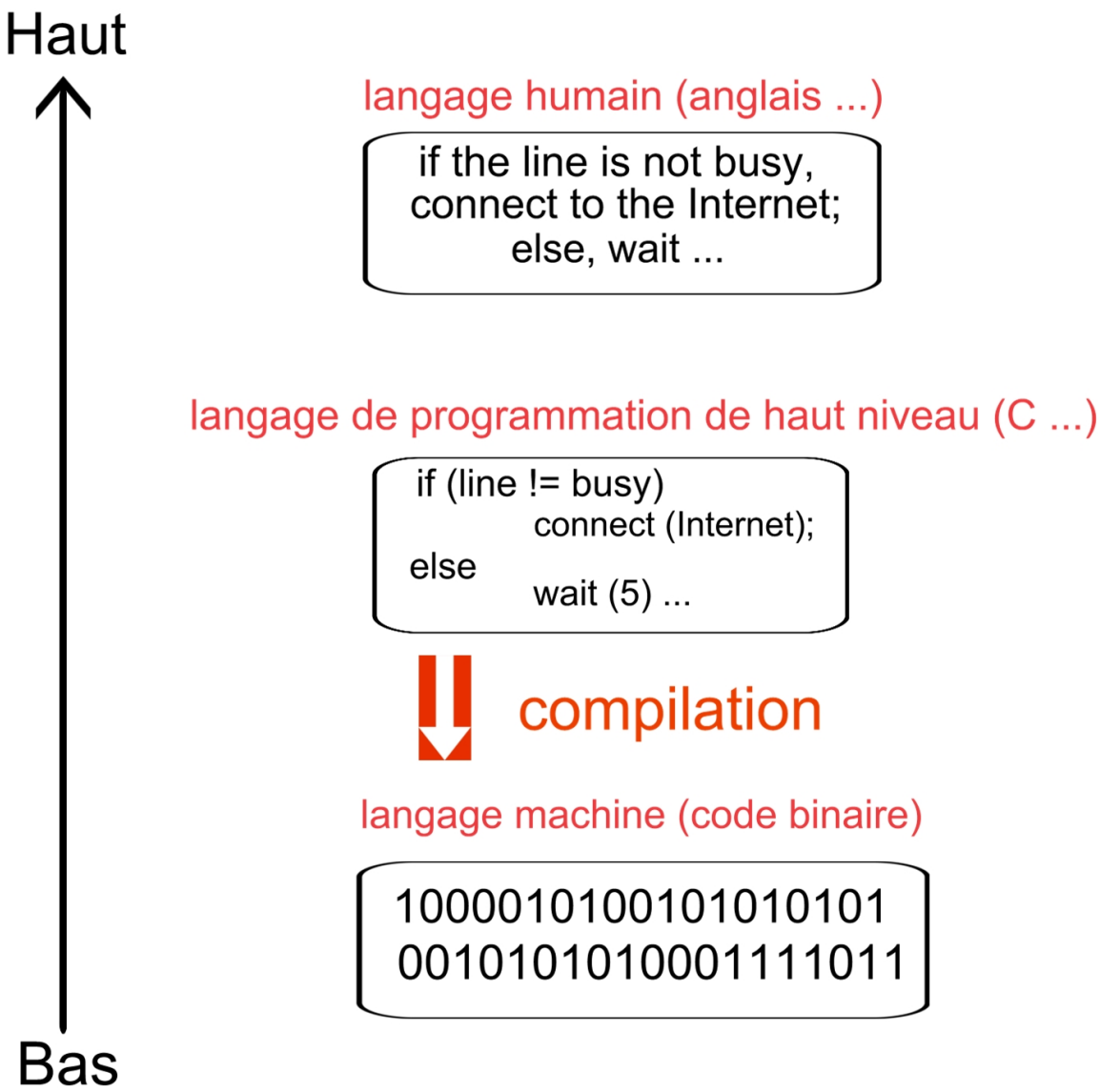
- Pourquoi le langage C ?

  $\longrightarrow$ un des langages de haut niveau les plus répandus dans le monde
  $\longrightarrow$ facile à lire et à comprendre, performant
  $\longrightarrow$ abordable car proche de l'anglais
  $\longrightarrow$ compilable sur tous les systèmes UNIX (Mandriva, RedHat, Mac OS, Ubuntu, Debian, Mandrake ...)
  $\longrightarrow$ maîtrise du C : Perl, C++, Java ... facile à apprendre

- Mais ...

  $\longrightarrow$ langage exigeant et explicite (permet d'être rigoureux)
  $\longrightarrow$ ... encore un langage à apprendre !

Niveaux logiques des langages



Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Pour utiliser ce programme il faut le compiler
  $\longrightarrow$ traduction du code en langage machine (0 et 1)

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Pour utiliser ce programme il faut le compiler
  $\longrightarrow$ traduction du code en langage machine (0 et 1)
- Utiliser une console dans le répertoire contenant le programme

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Pour utiliser ce programme il faut le compiler
  $\longrightarrow$ traduction du code en langage machine (0 et 1)
- Utiliser une console dans le répertoire contenant le programme
  ▷ gcc nom.du.programme

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Pour utiliser ce programme il faut le compiler
  $\longrightarrow$ traduction du code en langage machine (0 et 1)
- Utiliser une console dans le répertoire contenant le programme
  ▷ gcc nom.du.programme
  $\longrightarrow$ un fichier exécutable est créé : a.out (par défaut)

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Pour utiliser ce programme il faut le compiler
  $\longrightarrow$ traduction du code en langage machine (0 et 1)
- Utiliser une console dans le répertoire contenant le programme
  ▷ gcc nom.du.programme
  $\longrightarrow$ un fichier exécutable est créé : a.out (par défaut)
  $\longrightarrow$ pour exécuter a.out : ▷ ./a.out

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Que donne ce programme ???

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Que donne ce programme ???
  $\longrightarrow$ il imprime dans la console : vive la physique

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Que donne ce programme ???
  $\longrightarrow$ il imprime dans la console : vive la physique

********** explications **********

Un programme élémentaire

dans un fichier texte

```
1:    // Premier exemple de programme simple
2:    #include <stdio.h>
3:    int main()
4:    {
5:        printf("vive la physique\n");
6:        return 0;
7:    }
```

- Que donne ce programme ???
  $\longrightarrow$ il imprime dans la console : vive la physique

  ********** explications **********

- La ligne 1 contient un commentaire : // commentaire

Un programme élémentaire

dans un fichier texte

```
1:    // Premier exemple de programme simple
2:    #include <stdio.h>
3:    int main()
4:    {
5:        printf("vive la physique\n");
6:        return 0;
7:    }
```

- Que donne ce programme ???
  $\longrightarrow$ il imprime dans la console : vive la physique

  ********** explications **********

- La ligne 1 contient un commentaire : // commentaire
  $\longrightarrow$ le compilateur ne traite pas cette ligne (voir progs. 1, 2 et 3)

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Ligne 2 :
  $\longrightarrow$ #include : directive, ordre au préprocesseur (va chercher ...)

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Ligne 2 :
  $\longrightarrow$ #include : directive, ordre au préprocesseur (va chercher ...)
  $\longrightarrow$ < ... > : indication, va hors du répertoire actif

Un programme élémentaire

dans un fichier texte

```
1:    // Premier exemple de programme simple
2:    #include <stdio.h>
3:    int main()
4:    {
5:        printf("vive la physique\n");
6:        return 0;
7:    }
```

- Ligne 2 :
  $\longrightarrow$ #include : directive, ordre au préprocesseur (va chercher ...)
  $\longrightarrow$ < ... > : indication, va hors du répertoire actif
  $\longrightarrow$ stdio.h (STanDard Input/Output Header): fichier 'header' (en-tête) de la bibliothèque standard du C, toujours nécessaire

Un programme élémentaire

dans un fichier texte

```
1:    // Premier exemple de programme simple
2:    #include <stdio.h>
3:    int main()
4:    {
5:        printf("vive la physique\n");
6:        return 0;
7:    }
```

- Ligne 2 :
  → #include : directive, ordre au préprocesseur (va chercher ...)
  → < ... > : indication, va hors du répertoire actif
  → stdio.h (STanDard Input/Output Header): fichier 'header' (en-tête) de la bibliothèque standard du C, toujours nécessaire
  → Rq : on utilisera parfois d'autres fichiers en-tête (math.h, sdtlib.h, ..)

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Lignes 3 à 7 :
  $\longrightarrow$ main() : fonction principale du programme

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Lignes 3 à 7 :
  $\longrightarrow$ main() : fonction principale du programme
  $\longrightarrow$ fin du programme C lorsque toutes les instructions de main() sont traitées

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Lignes 3 à 7 :
  $\longrightarrow$ main() : fonction principale du programme
  $\longrightarrow$ fin du programme C lorsque toutes les instructions de main() sont traitées
- Ligne 5 :
  $\longrightarrow$ instruction printf() : imprime le contenu des guillemets

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Lignes 3 à 7 :
  $\longrightarrow$ main() : fonction principale du programme
  $\longrightarrow$ fin du programme C lorsque toutes les instructions de main() sont traitées
- Ligne 5 :
  $\longrightarrow$ instruction printf() : imprime le contenu des guillemets
  $\longrightarrow$ \n : retour chariot, saut de ligne

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

- Ligne 6 :
  $\longrightarrow$ return 0 : main renvoie 0 si tout c'est bien passé

Un programme élémentaire

dans un fichier texte

```
1:      // Premier exemple de programme simple
2:      #include <stdio.h>
3:      int main()
4:      {
5:          printf("vive la physique\n");
6:          return 0;
7:      }
```

En rouge : INDISPENSABLE dans tous vos programmes !

programme_part1_4.c

```
1:    #include <stdio.h>
2:    int main()
3:    {
4:        printf("\n \n");
5:        printf("**********************************************\n");
6:        printf("**            Laurent de FORGES             **\n");
7:        printf("**                                          **\n");
8:        printf("**       Institut Non Lineaire de Nice      **\n");
9:        printf("**   Universite de Nice Sophia-Antipolis    **\n");
10:       printf("**********************************************\n");
11:       printf("\n \n");
12:       return 0;
13:    }
```

- Ce programme donne :

```
**********************************************
**            Laurent de FORGES             **
**                                          **
**       Institut Non Lineaire de Nice      **
**   Universite de Nice Sophia-Antipolis    **
**********************************************
```

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant
- A quoi sert la directive #include ?

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant
- A quoi sert la directive #include ?
  $\longrightarrow$ elle ordonne au préprocesseur de chercher des fichiers et d'en associer le contenu au fichier .c

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant
- A quoi sert la directive #include ?
  $\longrightarrow$ elle ordonne au préprocesseur de chercher des fichiers et d'en associer le contenu au fichier .c
- A quoi sert la compilation ?

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant
- A quoi sert la directive #include ?
  $\longrightarrow$ elle ordonne au préprocesseur de chercher des fichiers et d'en associer le contenu au fichier .c
- A quoi sert la compilation ?
  $\longrightarrow$ l'ordinateur est incapable de comprendre directement un programme C, il faut donc le compiler pour le transformer en fichier exécutable

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant
- A quoi sert la directive #include ?
  $\longrightarrow$ elle ordonne au préprocesseur de chercher des fichiers et d'en associer le contenu au fichier .c
- A quoi sert la compilation ?
  $\longrightarrow$ l'ordinateur est incapable de comprendre directement un programme C, il faut donc le compiler pour le transformer en fichier exécutable
- Quel type de fichier produit le compilateur ?

Questions-réponses parties1

- A quoi servent les commentaires dans les programmes ?
  $\longrightarrow$ à documenter, indispensables dans les programmes complexes
- Quel est le rôle de la fonction main ?
  $\longrightarrow$ elle définit le début et la fin du programme, en son absence, le programme est inopérant
- A quoi sert la directive #include ?
  $\longrightarrow$ elle ordonne au préprocesseur de chercher des fichiers et d'en associer le contenu au fichier .c
- A quoi sert la compilation ?
  $\longrightarrow$ l'ordinateur est incapable de comprendre directement un programme C, il faut donc le compiler pour le transformer en fichier exécutable
- Quel type de fichier produit le compilateur ?
  $\longrightarrow$ un fichier exécutable (a.out par défaut)

Les variables en C

$\longrightarrow$ zones mémoires utilisées pour réserver des données

Les variables en C

$\longrightarrow$ zones mémoires utilisées pour réserver des données

$\longrightarrow$ différents types de données : nombres ou caractères alphabétiques

Les variables en C

$\longrightarrow$ zones mémoires utilisées pour réserver des données
$\longrightarrow$ différents types de données : nombres ou caractères alphabétiques
$\longrightarrow$ nombres : on distinguera les entiers de tous les autres (décimaux)

Les variables en C

$\longrightarrow$ zones mémoires utilisées pour réserver des données
$\longrightarrow$ différents types de données : nombres ou caractères alphabétiques
$\longrightarrow$ nombres : on distinguera les entiers de tous les autres (décimaux)

- Déclarer une variable (i) pour stocker un nombre entier : int i;

Les variables en C

$\longrightarrow$ zones mémoires utilisées pour réserver des données
$\longrightarrow$ différents types de données : nombres ou caractères alphabétiques
$\longrightarrow$ nombres : on distinguera les entiers de tous les autres (décimaux)

- Déclarer une variable (i) pour stocker un nombre entier : `int i;`
- Déclarer une variable (x) pour stocker un nombre décimale :

Les variables en C

⟶ zones mémoires utilisées pour réserver des données
⟶ différents types de données : nombres ou caractères alphabétiques
⟶ nombres : on distinguera les entiers de tous les autres (décimaux)

- Déclarer une variable (i) pour stocker un nombre entier : int i;
- Déclarer une variable (x) pour stocker un nombre décimale :

  ⟶ float x; (stocké sur 40 octets)

Les variables en C

⟶ zones mémoires utilisées pour réserver des données
⟶ différents types de données : nombres ou caractères alphabétiques
⟶ nombres : on distinguera les entiers de tous les autres (décimaux)

- Déclarer une variable (i) pour stocker un nombre entier : int i;
- Déclarer une variable (x) pour stocker un nombre décimale :

  ⟶ float x; (stocké sur 40 octets)
  ⟶ double x; (stocké sur 80 octets) ⟶ plus précis !

Les variables en C

$\longrightarrow$ zones mémoires utilisées pour réserver des données
$\longrightarrow$ différents types de données : nombres ou caractères alphabétiques
$\longrightarrow$ nombres : on distinguera les entiers de tous les autres (décimaux)

- Déclarer une variable (i) pour stocker un nombre entier : int i;
- Déclarer une variable (x) pour stocker un nombre décimale :

  $\longrightarrow$ float x; (stocké sur 40 octets)
  $\longrightarrow$ double x; (stocké sur 80 octets) $\longrightarrow$ plus précis !

- Déclarer une variable (c) pour stocker un caractère : char c;

programme_ part2_1.c

```
1:    #include <stdio.h>
2:    int main()
3:    {
4:        // declaration des variables
5:        int i;
6:        float x;
7:        double y;
8:        char c;
9:
10:       // affectations des variables
11:       i=2;
12:       x=4.2;
13:       y=3.14159;
14:       c='a';
15:
16:       // impression des variables
17:       printf("i vaut : %d\n",i);
18:       printf("x vaut : %f\n",x);
19:       printf("y vaut : %lf\n",y);
20:       printf("c vaut : %c\n",c);
22:   return 0;
23:   }
```

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d\n",i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Premier bloc : on déclare les variables
  $\longrightarrow$ un entier (i), deux décimaux (x,y) et un caractère (c)

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d\n",i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Deuxième bloc : on donne des valeurs aux variables
  $\longrightarrow$ attention : int=entier, float et double=décimaux, char=caractère

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d\n",i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Troisième bloc : on imprime les valeurs des variables
  $\longrightarrow$ on utilise l'instruction printf();

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d\n" ,i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Troisième bloc : on imprime les valeurs des variables
  $\longrightarrow$ entre parenthèses : texte à imprimer (délimité par les guillemets)

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d \n",i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Troisième bloc : on imprime les valeurs des variables
  $\longrightarrow$ entre parenthèses : indicateur du format de la variable (i)

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d \n",i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Troisième bloc : on imprime les valeurs des variables
  $\longrightarrow$ entre parenthèses : retour chariot, on passe à la ligne suivante

programme_part2_1.c (dans main)

```
4:      // declaration des variables
5:      int i;
6:      float x;
7:      double y;
8:      char c;
9:
10:     // affectations des variables
11:     i=2;
12:     x=4.2;
13:     y=3.14159;
14:     c='a';
15:
16:     // impression des variables
17:     printf("i vaut : %d\n",i);
18:     printf("x vaut : %f\n",x);
19:     printf("y vaut : %lf\n",y);
20:     printf("c vaut : %c\n",c);
```

- Troisième bloc : on imprime les valeurs des variables
  $\longrightarrow$ entre parenthèses : variable à afficher là ou se trouve l'indicateur (%d)

Les indicateurs de formats

- Caractère (char) :
  $\longrightarrow$ indicateur du format caractère : %c

Les indicateurs de formats

- Caractère (char) :
  $\longrightarrow$ indicateur du format caractère : %c
- Nombre entier (int) :
  $\longrightarrow$ indicateur du format entier : %d (ou %i)

Les indicateurs de formats

- Caractère (char) :
  $\longrightarrow$ indicateur du format caractère : %c
- Nombre entier (int) :
  $\longrightarrow$ indicateur du format entier : %d (ou %i)
- Nombre décimale (float) :
  $\longrightarrow$ indicateur du format float : %f

Les indicateurs de formats

- Caractère (char) :
  $\longrightarrow$ indicateur du format caractère : %c
- Nombre entier (int) :
  $\longrightarrow$ indicateur du format entier : %d (ou %i)
- Nombre décimale (float) :
  $\longrightarrow$ indicateur du format float : %f
- Nombre décimale (double) :
  $\longrightarrow$ indicateur du format double :%g (ou %lf)

Les indicateurs de formats : attention erreurs !

programme_part2_2.c (dans main)

```
4:      // declaration des variables
5:      int i, j;
6:
7:      // affectations des variables
8:      i=2, j=5;
9:
10:     // impression des variables
11:     printf("i vaut : %c\n",i);
12:     printf("j/i vaut : %i\n",j/i);
13:     printf("j/i vaut : %f\n",j/i);
```

- Cherchez les erreurs ...

Les indicateurs de formats : attention erreurs !

programme_part2_2.c (dans main)

```
4:      // declaration des variables
5:      int i, j;
6:
7:      // affectations des variables
8:      i=2, j=5;
9:
10:     // impression des variables
11:     printf("i vaut : %c\n",i);
12:     printf("j/i vaut : %i\n",j/i);
13:     printf("j/i vaut : %f\n",j/i);
```

- Cherchez les erreurs ...
  $\longrightarrow$ utilisation de l'indicateur de format de caractère %c avec une variable int : pas beau !

Les indicateurs de formats : attention erreurs !

programme_part2_2.c (dans main)

```
4:      // declaration des variables
5:      int i, j;
6:
7:      // affectations des variables
8:      i=2, j=5;
9:
10:     // impression des variables
11:     printf("i vaut : %c\n",i);
12:     printf("j/i vaut : %i\n",j/i);
13:     printf("j/i vaut : %f\n",j/i);
```

- Cherchez les erreurs ...
  $\longrightarrow$ délicat : %i appelle ici la division entière de $5/2$, soit 2. Il faut faire attention avec ce genre de subtilité !

Les indicateurs de formats : attention erreurs !

programme_part2_2.c (dans main)

```
4:      // declaration des variables
5:      int i, j;
6:
7:      // affectations des variables
8:      i=2, j=5;
9:
10:     // impression des variables
11:     printf("i vaut : %c\n",i);
12:     printf("j/i vaut : %i\n",j/i);
13:     printf("j/i vaut : %f\n",j/i);
```

- Cherchez les erreurs ...
  $\longrightarrow$ j/i est considéré comme un int car division de deux int. Utiliser %f va donc donner n'importe quoi : très sale !

Les indicateurs de formats : attention erreurs !

Une correction possible ...

programme_part2_2_corrige.c (dans main)

```
4:          // declaration des variables
5:          int i, j;
6:
7:          // affectations des variables
8:          i=2, j=5;
9:
10:         // impression des variables
11:         printf("i vaut : %d\n",i);
12:         printf("j/i vaut : %i\n",j/i);
13:         printf("j/i vaut : %f\n",(float) j/i);
```

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;
  $\longrightarrow$ sinon : const float x=4.111;

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;
  $\longrightarrow$ sinon : const float x=4.111;

- Déclaration des constantes avec un type implicitement précisé :

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;
  $\longrightarrow$ sinon : const float x=4.111;

- Déclaration des constantes avec un type implicitement précisé :
  $\longrightarrow$ #define PI 3.141592

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;
  $\longrightarrow$ sinon : const float x=4.111;

- Déclaration des constantes avec un type implicitement précisé :
  $\longrightarrow$ #define PI 3.141592
  $\longrightarrow$ #define N 10

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;
  $\longrightarrow$ sinon : const float x=4.111;

- Déclaration des constantes avec un type implicitement précisé :
  $\longrightarrow$ #define PI 3.141592
  $\longrightarrow$ #define N 10
  $\longrightarrow$ rq : pas de point virgule à la fin avec #define

Avant la fonction main()

- Déclaration des constantes avec un type explicitement précisé :
  $\longrightarrow$ pour les entiers : const int i=3;
  $\longrightarrow$ sinon : const float x=4.111;

- Déclaration des constantes avec un type implicitement précisé :
  $\longrightarrow$ #define PI 3.141592
  $\longrightarrow$ #define N 10
  $\longrightarrow$ rq : pas de point virgule à la fin avec #define

- voir : programme_part2_3.c

Questions-réponses parties 2

- Comment déclarer deux variables entières ?

Questions-réponses parties 2

- Comment déclarer deux variables entières ?
  $\longrightarrow$ int variable1, variable2;

Questions-réponses parties 2

- Comment déclarer deux variables entières ?

  $\longrightarrow$ int variable1, variable2;

  ou

  $\longrightarrow$ int variable1;

  $\longrightarrow$ int variable2;

Questions-réponses parties 2

- Comment déclarer deux variables entières ?
  $\longrightarrow$ int variable1, variable2;
  ou
  $\longrightarrow$ int variable1;
  $\longrightarrow$ int variable2;

- Que sont %c, %d et %f ?

Questions-réponses parties 2

- Comment déclarer deux variables entières ?
  $\longrightarrow$ int variable1, variable2;
  ou
  $\longrightarrow$ int variable1;
  $\longrightarrow$ int variable2;

- Que sont %c, %d et %f ?
  $\longrightarrow$ indicateur de format, utilisés avec printf ... permet de traiter le format caractère (%c), entier (%d) et décimal (%f)

Questions-réponses parties 2

- Comment déclarer deux variables entières ?
  $\longrightarrow$ int variable1, variable2;
  ou
  $\longrightarrow$ int variable1;
  $\longrightarrow$ int variable2;

- Que sont %c, %d et %f ?
  $\longrightarrow$ indicateur de format, utilisés avec printf ... permet de traiter le format caractère (%c), entier (%d) et décimal (%f)

- Les noms de variables suivants sont-ils valides ?

Questions-réponses parties 2

- Comment déclarer deux variables entières ?
  $\longrightarrow$ int variable1, variable2;
  ou
  $\longrightarrow$ int variable1;
  $\longrightarrow$ int variable2;

- Que sont %c, %d et %f ?
  $\longrightarrow$ indicateur de format, utilisés avec printf ... permet de traiter le format caractère (%c), entier (%d) et décimal (%f)

- Les noms de variables suivants sont-ils valides ?
  2eme_variable
  position's_2
  _rotation
  Amplitude_1

Questions-réponses parties 2

- Comment déclarer deux variables entières ?
  → int variable1, variable2;
  ou
  → int variable1;
  → int variable2;

- Que sont %c, %d et %f ?
  → indicateur de format, utilisés avec printf ... permet de traiter le format caractère (%c), entier (%d) et décimal (%f)

- Les noms de variables suivants sont-ils valides ?

  2eme_variable → pas valide !
  position's_2 → pas valide !
  _rotation → valide
  Amplitude_1 → valide

Entrées-sorties, manipuler des données

Comment entrer dans la console deux nombres et faire des opérations avec ?

Entrées-sorties, manipuler des données

Comment entrer dans la console deux nombres et faire des opérations avec ?

- Pour les entrées : fonction scanf();

Entrées-sorties, manipuler des données

Comment entrer dans la console deux nombres et faire des opérations avec ?

- Pour les entrées : fonction scanf();
- Pour les sorties : fonction printf();

Entrées-sorties, manipuler des données

Comment entrer dans la console deux nombres et faire des opérations avec ?

- Pour les entrées : fonction scanf();
- Pour les sorties : fonction printf();

programme_part3_1.c (dans main)

```
4:          // declaration des variables
5:          int i, j;
6:
7:         // affectations des variables via la console
8:         printf("Entrez deux nombres entiers i et j:\n");
9:         scanf("%d%d",&i,&j);
10:
11:          // impression des variables
12:          printf("i+j=%d\n",i+j);
13:          printf("i-j=%d\n",i-j);
14:          printf("i*j=%d\n",i*j);
15:          printf("i/j=%f\n",(float)i/j);
```

programme_part3_1.c (dans main)

```
4:          // declaration des variables
5:          int i, j;
6:
7:          // affectations des variables via la console
8:          printf("Entrez deux nombres entiers i et j:\n");
9:          scanf("%d%d",&i,&j);
```

- scanf() permet d'affecter une valeur à une (ou plusieurs) variable(s)

programme_part3_1.c (dans main)

```
4:          // declaration des variables
5:          int i, j;
6:
7:         // affectations des variables via la console
8:         printf("Entrez deux nombres entiers i et j:\n");
9:         scanf("%d%d",&i,&j);
```

- scanf() permet d'affecter une valeur à une (ou plusieurs) variable(s)
- entre guillemets : type de format des variables (sans virgule)

programme_part3_1.c (dans main)

```
4:          // declaration des variables
5:          int i, j;
6:
7:          // affectations des variables via la console
8:          printf("Entrez deux nombres entiers i et j:\n");
9:          scanf("%d%d",&i,&j);
```

- scanf() permet d'affecter une valeur à une (ou plusieurs) variable(s)
- entre guillemets : type de format des variables (sans virgule)
- après la virgule : noms des variables (&: voir pointeurs)

Que se passe t-il si l'on entre un nombre décimale dans la console ???

programme_part3_1.c (dans main)

```
4:          // declaration des variables
5:          int i, j;
6:
7:         // affectations des variables via la console
8:         printf("Entrez deux nombres entiers i et j:\n");
9:         scanf("%d%d",&i,&j);
10:
11:          // impression des variables
12:          printf("i+j=%d\n",i+j);
13:          printf("i-j=%d\n",i-j);
14:          printf("i*j=%d\n",i*j);
15:          printf("i/j=%f\n",(float)i/j);
```

Plus fun !!!

programme_part3_2.c

```
1:      #include <stdio.h>
2:      int main()
3:      {
4:          // declaration des variables
5:          int i, j;
6:
7:         // affectations des variables via la console
8:         printf("Entre ton jour et moi de naissance :\n");
9:         scanf("%d%d",&i,&j);
10:
11:          // impression des variables
12:         printf("Il te reste %d jours a vivre !\n",i*j);
13:         printf("PROFITE UN MAX DE TA VIE !!!\n");
14:
15:      return 0;
16:      }
```

Encore plus fort

Encore plus fort

- Pour lire dans un fichier : fonction fscanf();

Encore plus fort

- Pour lire dans un fichier : fonction fscanf();
- Pour écrire dans un fichier : fonction fprintf();

Pour lire dans le fichier mesdonnees_input.txt avec la fonction fscanf()

fonction scanf (dans main)

```
1:      FILE *file1; // declaration d un pointeur de type FILE
2:
3:      double x, y, z; // declaration des variables (decimales)
4:
5:      // pour ouvrir le fichier existant mesdonnees_input.txt
6:      file1=fopen("mesdonnees_input.txt","r");
7:
8:      // pour affecter le contenu du fichier aux variables
9:      fscanf(file1,"%lf%lf%lf",&x,&y,&z);
10:
11:     // il faut refermer le fichier lu
12:     fclose(file1);
```

fonction scanf (dans main)

```
1:    FILE *file1; // declaration d un pointeur de type FILE
2:
3:    double x, y, z; // declaration des variables (decimales)
4:
5:    // pour ouvrir le fichier existant mesdonnees_input.txt
6:    file1=fopen("mesdonnees_input.txt","r");
7:
8:    // pour affecter le contenu du fichier aux variables
9:    fscanf(file1,"%lf%lf%lf",&x,&y,&z);
10:
11:   // il faut refermer le fichier lu
12:   fclose(file1);
```

→ FILE : structure de contrôle, *file1: pointeur de type FILE

fonction scanf (dans main)

```
1:      FILE *file1; // declaration d un pointeur de type FILE
2:
3:      double x, y, z; // declaration des variables (decimales)
4:
5:      // pour ouvrir le fichier existant mesdonnees_input.txt
6:      file1=fopen("mesdonnees_input.txt","r");
7:
8:      // pour affecter le contenu du fichier aux variables
9:      fscanf(file1,"%lf%lf%lf",&x,&y,&z);
10:
11:     // il faut refermer le fichier lu
12:     fclose(file1);
```

$\longrightarrow$ association du flux du fichier au pointeur file1

fonction scanf (dans main)

```
1:      FILE *file1; // declaration d un pointeur de type FILE
2:
3:      double x, y, z; // declaration des variables (decimales)
4:
5:      // pour ouvrir le fichier existant mesdonnees_input.txt
6:      file1=fopen("mesdonnees_input.txt","r");
7:
8:      // pour affecter le contenu du fichier aux variables
9:      fscanf(file1,"%lf%lf%lf",&x,&y,&z);
10:
11:     // il faut refermer le fichier lu
12:     fclose(file1);
```

→ fopen : ouvre le fichier mesdonnees_input.txt en mode lecture (r : read)

fonction scanf (dans main)

```
1:      FILE *file1; // declaration d un pointeur de type FILE
2:
3:      double x, y, z; // declaration des variables (decimales)
4:
5:      // pour ouvrir le fichier existant mesdonnees_input.txt
6:      file1=fopen("mesdonnees_input.txt","r");
7:
8:      // pour affecter le contenu du fichier aux variables
9:      fscanf(file1,"%lf%lf%lf",&x,&y,&z);
10:
11:     // il faut refermer le fichier lu
12:     fclose(file1);
```

$\longrightarrow$ le fichier input est ouvert, scanf() affecte ce qu'on lit aux variables x, y et z

fonction scanf (dans main)

```
1:      FILE *file1; // declaration d un pointeur de type FILE
2:
3:      double x, y, z; // declaration des variables (decimales)
4:
5:      // pour ouvrir le fichier existant mesdonnees_input.txt
6:      file1=fopen("mesdonnees_input.txt","r");
7:
8:      // pour affecter le contenu du fichier aux variables
9:      fscanf(file1,"%lf%lf%lf",&x,&y,&z);
10:
11:     // il faut refermer le fichier lu
12:     fclose(file1);
```

$\longrightarrow$ fclose sauve les données dans un buffer, ferme le fichier associé au flux (file1)

Pour écrire dans le fichier mesdonnees_output.txt avec la fonction fprintf()

fonction fprintf (dans main)

```
1:      FILE *file2; // declaration d un pointeur de type FILE
2:
3:     double x=2.1, y=4.0, z=5.4; // declaration et affectation des variables (decimales)
4:
5:      // pour creer le fichier mesdonnees_output.txt
6:      file2=fopen("mesdonnees_output.txt","w");
7:
8:      // pour imprimer dans le fichier mesdonnees_output.txt
9:      fprintf(file2,"%lf     %lf\n",x,y+z);
10:
11:     // il faut refermer le fichier créé
12:     fclose(file2);
```

fonction fprintf (dans main)

```
1:      FILE *file2; // declaration d un pointeur de type FILE
2:
3:     double x=2.1, y=4.0, z=5.4; // declaration et affectation des variables (decimales)
4:
5:      // pour creer le fichier mesdonnees_output.txt
6:      file2=fopen("mesdonnees_output.txt","w");
7:
8:      // pour imprimer dans le fichier mesdonnees_output.txt
9:      fprintf(file2,"%lf     %lf\n",x,y+z);
10:
11:     // il faut refermer le fichier créé
12:     fclose(file2);
```

→ fopen : crée le fichier mesdonnees_output.txt avec l'option écriture (w : write)

fonction fprintf (dans main)

```
1:      FILE *file2; // declaration d un pointeur de type FILE
2:
3:     double x=2.1, y=4.0, z=5.4; // declaration et affectation des variables (decimales)
4:
5:      // pour creer le fichier mesdonnees_output.txt
6:      file2=fopen("mesdonnees_output.txt","w");
7:
8:      // pour imprimer dans le fichier mesdonnees_output.txt
9:      fprintf(file2,"%lf     %lf\n",x,y+z);
10:
11:     // il faut refermer le fichier créé
12:     fclose(file2);
```

$\longrightarrow$ ici aussi, fclose sauve les données et ferme le fichier associé au flux (file2)

Lire dans un fichier, manipuler les données et écrire dans un autre fichier !

programme_part3_3.c (dans main) avec mesdonnees_input.txt

```
1:      FILE * file1;
2:      FILE * file2;
3:
4:      double x1, y1, z1;
5:      double x2, y2, z2;
6:
7:      file1=fopen("mesdonnees_input.txt","r");
8:      file2=fopen("mesdonnees_output.txt","w");
9:
10:      fscanf(file1,"%lf%lf%lf",&x1,&y1,&z1);
11:      fscanf(file1,"%lf%lf%lf",&x2,&y2,&z2);
12:
13:      fprintf(file2,"%lf      %lf\n",x1,y1+z1);
14:      fprintf(file2,"%lf      %lf\n",x2,y2+z2);
15:
16:      fclose(file1);
17:      fclose(file2);
```

En Physique (Licence 3)

En Physique (Licence 3)

- On utilisera surtout printf et fprintf

En Physique (Licence 3)

- On utilisera surtout printf et fprintf
- On calculera des quantités que l'on imprimera dans des fichiers

En Physique (Licence 3)

- On utilisera surtout printf et fprintf
- On calculera des quantités que l'on imprimera dans des fichiers
- Pour cela, on répétera des opérations grâce aux instructions

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte
- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants
- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'
- Corrigez les erreurs :

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'
- Corrigez les erreurs :
  fprintf("coucou\n");

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'

- Corrigez les erreurs :
  fprintf("coucou\n"); $\longrightarrow$ fprintf(file1,"coucou\n");

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'

- Corrigez les erreurs :
  fprintf("coucou\n"); $\longrightarrow$ fprintf(file1,"coucou\n");
  scanf("%d",i)

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'

- Corrigez les erreurs :
  fprintf("coucou\n"); $\longrightarrow$ fprintf(file1,"coucou\n");
  scanf("%d",i) $\longrightarrow$ scanf("%d",&i);

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'

- Corrigez les erreurs :
  fprintf("coucou\n"); $\longrightarrow$ fprintf(file1,"coucou\n");
  scanf("%d",i) $\longrightarrow$ scanf("%d",&i);
  fscanf(file1,"%d");

Questions-réponses parties 3

- A quoi servent les fonctions printf, fprintf, scanf et fscanf ?
  printf $\longrightarrow$ permet d'imprimer à l'écran (console)
  fprintf $\longrightarrow$ permet d'imprimer dans un fichier texte
  scanf $\longrightarrow$ permet de lire à l'écran (console)
  fscanf $\longrightarrow$ permet de lire dans un fichier texte

- J'utilise fprintf et j'omet fclose(file1); le programme peut-il être compilé ?
  $\longrightarrow$ non car il faut fermer les flux entrants et/ou sortants

- A quoi servent "r" et "w" dans programme_part3_gamma.c ?
  $\longrightarrow$ "r" pour 'read' et "w" 'pour write'

- Corrigez les erreurs :
  fprintf("coucou\n"); $\longrightarrow$ fprintf(file1,"coucou\n");
  scanf("%d",i) $\longrightarrow$ scanf("%d",&i);
  fscanf(file1,"%d"); $\longrightarrow$ fscanf(file1,"%d",&i);

Traitements répétitifs : for, while et do-while

Traitements répétitifs : for, while et do-while

- En Physique, il est très souvent utile de répéter une opération

Traitements répétitifs : for, while et do-while

- En Physique, il est très souvent utile de répéter une opération
- On va utiliser des boucles, des traitements répétitifs que le programme répète jusqu'à ce qu'une ou plusieurs conditions soient vérifiées

Traitements répétitifs : for, while et do-while

- En Physique, il est très souvent utile de répéter une opération
- On va utiliser des boucles, des traitements répétitifs que le programme répète jusqu'à ce qu'une ou plusieurs conditions soient vérifiées
- Ex : tracer la fonction gaussienne $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}$ sur l'intervalle $x \in [-5, +5]$ avec $\sigma = 1$

Tracer $f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ sur l'intervalle $x \in [-5, +5]$

programme_part4_1.c

```
1:    #include <stdio.h>
2:    #include <math.h>     // on inclut les librairies pour utiliser sqrt et exp
3:
4:    #define PI 3.141592  // declaration de la constante PI
5:
6:    int main()
7:    {
8:     ...
9:    return 0;
10:   }
```

Instructions et opérateurs logiques

Tracer $f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ sur l'intervalle $x \in [-5, +5]$

programme_part4_1.c

```
1:    #include <stdio.h>
2:    #include <math.h>     // on inclut les librairies pour utiliser sqrt et exp
3:
4:    #define PI 3.141592  // declaration de la constante PI
5:
6:    int main()
7:    {
8:     ...
9:    return 0;
10:   }
```

- Pour utiliser les fonctions mathématiques en C, on inclut math.h

Tracer $f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$ sur l'intervalle $x \in [-5, +5]$

programme_part4_1.c

```
1:    #include <stdio.h>
2:    #include <math.h>     // on inclut les librairies pour utiliser sqrt et exp
3:
4:    #define PI 3.141592  // declaration de la constante PI
5:
6:    int main()
7:    {
8:     ...
9:    return 0;
10:   }
```

- Pour utiliser les fonctions mathématiques en C, on inclut math.h
- Pour compiler, on rajoute -lm : ▷ gcc fichier.c -lm

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

→ Variables pour les bornes, la position et la norme de la gaussienne

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

→ La variable i va servir pour la boucle for

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

→ Nombrepoints est le nombre total de points que l'on va calculer

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ On donne des valeurs aux variables

Instructions et opérateurs logiques

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

→ Boucle qui répète son opération jusqu'à ce que i=Nombrepoints

Instructions et opérateurs logiques

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ i=0, opération, i=1, opération, i=2, opération, ... i=Nombrepoints, stop

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ Trois expressions dans le champ d'expression (.. ; .. ; ..)

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ i plus petit ou égal à Nombrepoints

Instructions et opérateurs logiques

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ i++ : incrémente i d'une unité à chaque nouvelle exécution

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ Au début de la boucle : i=0 $\Rightarrow$ x=xmin

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ A la fin de la boucle : i= Nombrepoints $\Rightarrow$ x=xmax

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ A un instant quelconque : δx= (xmax-xmin)/Nombrepoints

programme_part4_1.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:
9:      for(i=0;i<=Nombrepoints;i++)
10:     {
11:             x=xmin+i*(xmax-xmin)/Nombrepoints;
12:             printf("%f     %f\n",x,norm*exp(-x*x/2));
13:     }
```

$\longrightarrow$ A chaque étape, on imprime x et f(x) dans la console

Il est plus pratique d'imprimer x et f(x) dans un fichier texte avec fprintf

programme_part4_2.c (dans main)

```
1:    FILE * file;
2:    file=fopen("gauss_ouput.txt","w");
3:
4:    float xmin, xmax, x, norm;
5:    int i, Nombrepoints;
6:
7:    xmin=-5;
8:    xmax=5;
9:    norm=1/sqrt(2*PI);
10:   Nombrepoints=200;
11:
12:   for(i=0;i<=Nombrepoints;i++)
13:   {
14:         x=xmin+i*(xmax-xmin)/Nombrepoints;
15:         fprintf(file,"%f     %f\n",x,norm*exp(-x*x/2));
16:   }
17:   fclose(file);
```

Instructions et opérateurs logiques

Pour aller un peu plus loin ...

- On peut même entrer dans la console la valeur de σ

Instructions et opérateurs logiques

Pour aller un peu plus loin ...

- On peut même entrer dans la console la valeur de σ
- Un peu plus technique mais bien pratique !

Instructions et opérateurs logiques

Pour aller un peu plus loin ...

- On peut même entrer dans la console la valeur de σ
- Un peu plus technique mais bien pratique !
- Voir : programme_part4_3.c

On peut utiliser while à la place de for dans programme_part4_1.c

programme_part4_4.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:      i=0;
9:
10:     while(i<=Nombrepoints)
11:     {
12:             x=xmin+i*(xmax-xmin)/Nombrepoints;
13:             printf("%f     %f\n",x,norm*exp(-x*x/2));
14:             i = i+1;
15:     }
```

programme_part4_4.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:      i=0;
9:
10:     while(i<=Nombrepoints)
11:     {
12:             x=xmin+i*(xmax-xmin)/Nombrepoints;
13:             printf("%f     %f\n",x,norm*exp(-x*x/2));
14:             i = i+1;
15:     }
```

$\longrightarrow$ while = "as long as" en C = tant que

programme_part4_4.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:      i=0;
9:
10:     while(i<=Nombrepoints)
11:     {
12:             x=xmin+i*(xmax-xmin)/Nombrepoints;
13:             printf("%f     %f\n",x,norm*exp(-x*x/2));
14:             i = i+1;
15:     }
```

→ Avantage ou désavantage : une seule expression

Autre fonction utile : do-while

programme_part4_5.c (dans main)

```
1:      float xmin, xmax, x, norm;
2:      int i, Nombrepoints;
3:
4:      xmin=-5;
5:      xmax=5;
6:      norm=1/sqrt(2*PI);
7:      Nombrepoints=200;
8:      i=0;
9:
10:     do
11:     {
12:             x=xmin+i*(xmax-xmin)/Nombrepoints;
13:             printf("%f     %f\n",x,norm*exp(-x*x/2));
14:             i = i+1;
15:     }
16:     while(i<=Nombrepoints);
```

Boucle do-while :

- for et while : exécution des instructions conditionnée par le résultat de l'évaluation

Boucle do-while :

- for et while : exécution des instructions conditionnée par le résultat de l'évaluation
- Avec do-while : exécute d'abord les instructions puis test ensuite l'expression

Boucle do-while :

- for et while : exécution des instructions conditionnée par le résultat de l'évaluation
- Avec do-while : exécute d'abord les instructions puis test ensuite l'expression
- On exécutera toujours au moins une fois l'instruction

Autre avantage des boucles : elles peuvent s'imbriquer !

programme_part4_6.c

```
1:      #include <stdio.h>
2:      int main()
3:      {
4:              int i, j;
5:              int imax=7, jmax=5;
6:
7:              for(i=1;i<=imax;i++)
8:              {
9:                      for(j=1;j<=jmax;j++)
10:                     {
11:                         printf("c%d%d     ",i,j);
12:                     }
13:              printf("\n");
14:              }
15:     return 0;
16:     }
```

Soyez propre sinon vous êtes perdu ...

programme_part4_6.c

```
1:      #include <stdio.h>
2:      int main()
3:      {
4:              int i, j;
5:              int imax=7, jmax=5;
6:
7:              for(i=1;i<=imax;i++)
8:              {
9:                      for(j=1;j<=jmax;j++)
10:                     {
11:                         printf("c%d%d     ",i,j);
12:                     }
13:             printf("\n");
14:             }
15:     return 0;
16:     }
```

Soyez propre sinon vous êtes perdu ...

programme_part4_6.c

```
1:      #include <stdio.h>
2:      int main()
3:      {
4:              int i, j;
5:              int imax=7, jmax=5;
6:
7:              for(i=1;i<=imax;i++)
8:              {
9:                      for(j=1;j<=jmax;j++)
10:                     {
11:                         printf("c%d%d     ",i,j);
12:                     }
13:             printf("\n");
14:             }
15:     return 0;
16:     }
```

Soyez propre sinon vous êtes perdu ...

programme_part4_6.c

```
1:      #include <stdio.h>
2:      int main()
3:      {
4:              int i, j;
5:              int imax=7, jmax=5;
6:
7:              for(i=1;i<=imax;i++)
8:              {
9:                      for(j=1;j<=jmax;j++)
10:                     {
11:                             printf("c%d%d     ",i,j);
12:                     }
13:             printf("\n");
14:             }
15:     return 0;
16:     }
```

Exécuter une instruction sous condition : if

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:            int i,N;
7:            srand48(time(NULL));
8:            i=10*drand48();
9:
10:           printf("Entrez un nombre entre 0 et 10 :\n");
11:           scanf("%d",&N);
12:           printf("i=%d\n",i);
13:
14:           if(N==i)
15:           {
16:                   printf("WAW ... VOUS AVEZ GAGNE\n");
17:           }
18:    return 0;
19:    }
```

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:            int i,N;
7:            srand48(time(NULL));
8:            i=10*drand48();
9:
10:           printf("Entrez un nombre entre 0 et 10 :\n");
11:           scanf("%d",&N);
12:           printf("i=%d\n",i);
13:
14:           if(N==i)
15:           {
16:                   printf("WAW ... VOUS AVEZ GAGNE\n");
17:           }
18:    return 0;
19:    }
```

→ Librairies nécessaires pour le générateur aléatoire

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:          int i,N;
7:          srand48(time(NULL));
8:          i=10*drand48();
9:
10:         printf("Entrez un nombre entre 0 et 10 :\n");
11:         scanf("%d",&N);
12:         printf("i=%d\n",i);
13:
14:         if(N==i)
15:         {
16:               printf("WAW ... VOUS AVEZ GAGNE\n");
17:         }
18:    return 0;
19:    }
```

→ Permet de tirer un nombre décimale aléatoire compris entre 0 et 1

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:            int i,N;
7:            srand48(time(NULL));
8:            i=10*drand48();
9:
10:           printf("Entrez un nombre entre 0 et 10 :\n");
11:           scanf("%d",&N);
12:           printf("i=%d\n",i);
13:
14:           if(N==i)
15:           {
16:                   printf("WAW ... VOUS AVEZ GAGNE\n");
17:           }
18:    return 0;
19:    }
```

→ On en fait un nombre entier compris entre 0 et 10

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:            int i,N;
7:            srand48(time(NULL));
8:            i=10*drand48();
9:
10:           printf("Entrez un nombre entre 0 et 10 :\n");
11:           scanf("%d",&N);
12:           printf("i=%d\n",i);
13:
14:           if(N==i)
15:           {
16:                   printf("WAW ... VOUS AVEZ GAGNE\n");
17:           }
18:    return 0;
19:    }
```

→ Le nombre entré par utilisateur dans la console est lu

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:            int i,N;
7:            srand48(time(NULL));
8:            i=10*drand48();
9:
10:           printf("Entrez un nombre entre 0 et 10 :\n");
11:           scanf("%d",&N);
12:           printf("i=%d\n",i);
13:
14:           if(N==i)
15:           {
16:                   printf("WAW ... VOUS AVEZ GAGNE\n");
17:           }
18:    return 0;
19:    }
```

→ On imprime la valeur de i pour vérification

programme_part4_7.c

```
1:    #include <stdio.h>
2:    #include <stdlib.h>
3:    #include <time.h>
4:    int main()
5:    {
6:            int i,N;
7:            srand48(time(NULL));
8:            i=10*drand48();
9:
10:           printf("Entrez un nombre entre 0 et 10 :\n");
11:           scanf("%d",&N);
12:           printf("i=%d\n",i);
13:
14:           if(N==i)
15:           {
16:                   printf("WAW ... VOUS AVEZ GAGNE\n");
17:           }
18:    return 0;
19:    }
```

→ Si i et N sont égaux, alors l'utilisateur à gagné : on imprime le texte

Pour aller encore plus loin : if-else

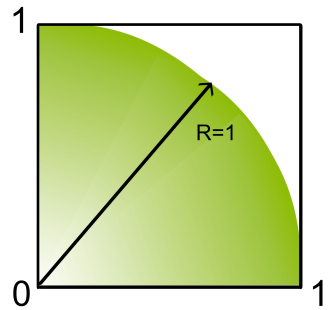
programme_part4_8.c (dans main)

```
1:      int i,N;
2:      srand48(time(NULL));
3:      i=10*drand48();
4:
5:      printf("Entrez un nombre entre 0 et 10 :\n");
6:      scanf("%d",&N);
7:      printf("i=%d\n",i);
8:
9:      if(N==i)
10:     {
11:             printf("WAW ... VOUS AVEZ GAGNE\n");
12:     }
13:     else
14:     {
15:             printf("PERDU ... RECOMMENCEZ\n");
16:     }
```

Créer un programme qui calcule π

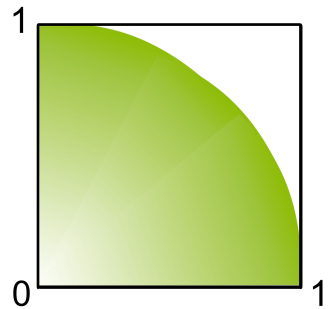
- Utiliser la géométrie et la loi des grands nombres
- Géométrie :

  Aire du carré : $\mathcal{A}_{car} = 1$
  Aire du quart de cercle : $\mathcal{A}_{cer} = \pi/4$



- Loi des grands nombres :

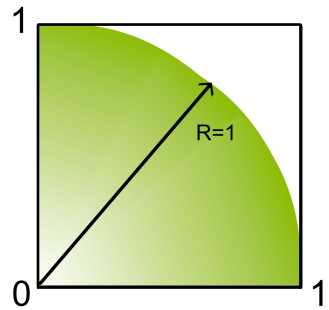
  On tire N$\gg$1 points aléatoirement
  dans le carré



Créer un programme qui calcule π

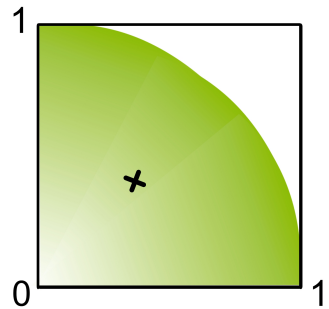
- Utiliser la géométrie et la loi des grands nombres
- Géométrie :

  Aire du carré : $\mathcal{A}_{car} = 1$
  Aire du quart de cercle : $\mathcal{A}_{cer} = \pi/4$



- Loi des grands nombres :

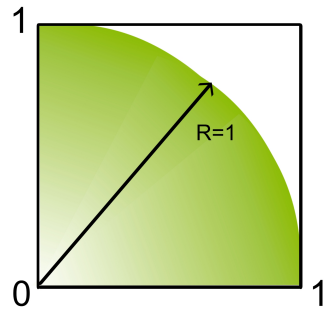
  On tire N$\gg$1 points aléatoirement
  dans le carré



Créer un programme qui calcule π

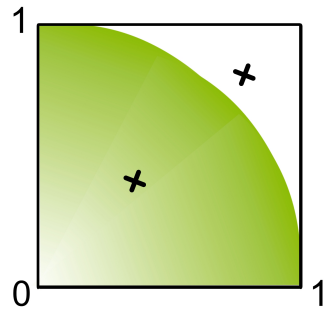
- Utiliser la géométrie et la loi des grands nombres
- Géométrie :

  Aire du carré : $\mathcal{A}_{car} = 1$
  Aire du quart de cercle : $\mathcal{A}_{cer} = \pi/4$



- Loi des grands nombres :

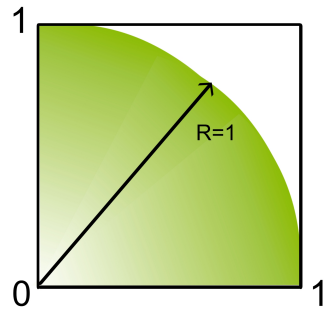
  On tire N$\gg$1 points aléatoirement
  dans le carré



Créer un programme qui calcule π

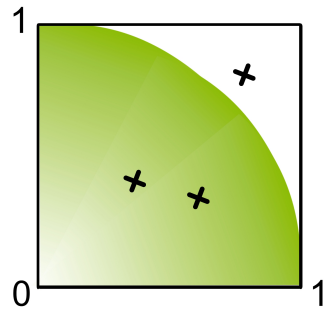
- Utiliser la géométrie et la loi des grands nombres
- Géométrie :

  Aire du carré : $\mathcal{A}_{car} = 1$
  Aire du quart de cercle : $\mathcal{A}_{cer} = \pi/4$



- Loi des grands nombres :

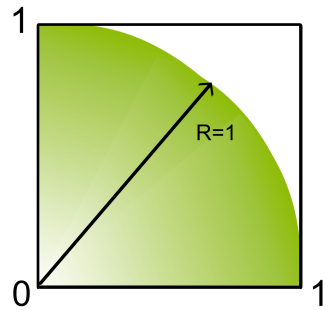
  On tire N$\gg$1 points aléatoirement
  dans le carré



Créer un programme qui calcule π

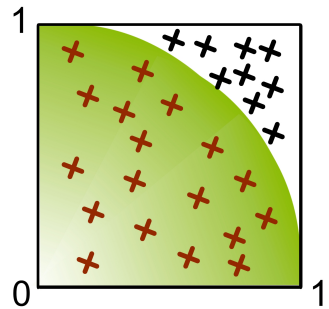
- Utiliser la géométrie et la loi des grands nombres
- Géométrie :

  Aire du carré : $\mathcal{A}_{car} = 1$
  Aire du quart de cercle : $\mathcal{A}_{cer} = \pi/4$



- Loi des grands nombres :

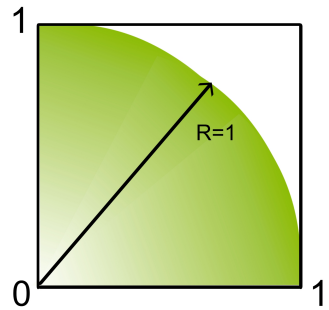
  On compte le nombre de croix dans le quart de cercle : n



Créer un programme qui calcule π

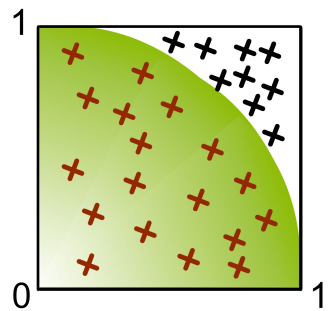
- Utiliser la géométrie et la loi des grands nombres
- Géométrie :

  Aire du carré : $\mathcal{A}_{car} = 1$
  Aire du quart de cercle : $\mathcal{A}_{cer} = \pi/4$



- Loi des grands nombres :

$$\lim_{N \to \infty} \frac{n}{N} = \frac{\mathcal{A}_{cer}}{\mathcal{A}_{car}} = \frac{\pi}{4}$$



programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

$\longrightarrow$ x, y : positions du point tiré, R : distance entre le point et l'origine

programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

$\longrightarrow$ i : variable muette pour la boucle for

Instructions et opérateurs logiques

programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

$\longrightarrow$ n : nombre de points compris dans le quart de cercle

programme_part4_9.c (dans main)

```
1:          double x, y, R, Pi;
2:          int i, n, N;
3:          srand48(time(NULL));
4:
5:          N=100000000;
6:          n=0;
7:
8:          for(i=0;i<N;i++)
9:          {
10:                 x=drand48();
11:                 y=drand48();
12:                 R=sqrt(x*x+y*y);
13:
14:                 if(R<=1)
15:                 {
16:                         n++;
17:                 }
18:           }
19:           Pi=4*(float)n/(float)N;
20:           printf("Pi=%f\n",Pi);
```

$\longrightarrow$ N : nombre total de points tirés

programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

→ Pour utiliser le générateur de nombre aléatoire

programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

$\longrightarrow$ Initialisation des variables N et n

Instructions et opérateurs logiques

programme_part4_9.c (dans main)

```
1:          double x, y, R, Pi;
2:          int i, n, N;
3:          srand48(time(NULL));
4:
5:          N=100000000;
6:          n=0;
7:
8:          for(i=0;i<N;i++)
9:          {
10:                 x=drand48();
11:                 y=drand48();
12:                 R=sqrt(x*x+y*y);
13:
14:                 if(R<=1)
15:                 {
16:                         n++;
17:                 }
18:           }
19:           Pi=4*(float)n/(float)N;
20:           printf("Pi=%f\n",Pi);
```

$\longrightarrow$ i vaut 0, on exécute les instructions en incrémentant i de 1 à chaque étape

Instructions et opérateurs logiques

programme_part4_9.c (dans main)

```
1:          double x, y, R, Pi;
2:          int i, n, N;
3:          srand48(time(NULL));
4:
5:          N=100000000;
6:          n=0;
7:
8:          for(i=0;i<N;i++)
9:          {
10:                 x=drand48();
11:                 y=drand48();
12:                 R=sqrt(x*x+y*y);
13:
14:                 if(R<=1)
15:                 {
16:                         n++;
17:                 }
18:           }
19:           Pi=4*(float)n/(float)N;
20:           printf("Pi=%f\n",Pi);
```

→ On s'arrête lorsque i vaut N-1 (< : strictement inférieur à ...)

programme_part4_9.c (dans main)

```
1:          double x, y, R, Pi;
2:          int i, n, N;
3:          srand48(time(NULL));
4:
5:          N=100000000;
6:          n=0;
7:
8:          for(i=0;i<N;i++)
9:          {
10:                 x=drand48();
11:                 y=drand48();
12:                 R=sqrt(x*x+y*y);
13:
14:                 if(R<=1)
15:                 {
16:                         n++;
17:                 }
18:           }
19:           Pi=4*(float)n/(float)N;
20:           printf("Pi=%f\n",Pi);
```

$\longrightarrow$ On tire un nombre dans le carré, R : distance origine-point

Instructions et opérateurs logiques

programme_part4_9.c (dans main)

```
1:          double x, y, R, Pi;
2:          int i, n, N;
3:          srand48(time(NULL));
4:
5:          N=100000000;
6:          n=0;
7:
8:          for(i=0;i<N;i++)
9:          {
10:                 x=drand48();
11:                 y=drand48();
12:                 R=sqrt(x*x+y*y);
13:
14:                 if(R<=1)
15:                 {
16:                         n++;
17:                 }
18:           }
19:           Pi=4*(float)n/(float)N;
20:           printf("Pi=%f\n",Pi);
```

$\longrightarrow$ Si le point se situe dans le quart de cercle, on rajoute 1 à la valeur de n

Instructions et opérateurs logiques

programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

$\longrightarrow$ On calcule la valeur de π en faisant attention à la conversion décimale

programme_part4_9.c (dans main)

```
1:      double x, y, R, Pi;
2:      int i, n, N;
3:      srand48(time(NULL));
4:
5:      N=100000000;
6:      n=0;
7:
8:      for(i=0;i<N;i++)
9:      {
10:             x=drand48();
11:             y=drand48();
12:             R=sqrt(x*x+y*y);
13:
14:             if(R<=1)
15:             {
16:                     n++;
17:             }
18:       }
19:       Pi=4*(float)n/(float)N;
20:       printf("Pi=%f\n",Pi);
```

$\longrightarrow$ On imprime la valeur de π dans la console

Les opérateurs relationnels

- == : égal à

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à
- $<$: inférieur à

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à
- $<$: inférieur à
- $>=$: supérieur ou égal à

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à
- $<$: inférieur à
- $>=$: supérieur ou égal à
- $<=$: inférieur ou égal à

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à
- $<$: inférieur à
- $>=$: supérieur ou égal à
- $<=$: inférieur ou égal à

Les opérateurs logiques

- && : ET (donne 1 si les expressions sont simultanément réalisées)

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à
- $<$: inférieur à
- $>=$: supérieur ou égal à
- $<=$: inférieur ou égal à

Les opérateurs logiques

- && : ET (donne 1 si les expressions sont simultanément réalisées)
- || : OU (inclusif : donne 1 si au moins une des expressions est réalisée)

Les opérateurs relationnels

- $==$: égal à
- $!=$: différent de
- $>$: supérieur à
- $<$: inférieur à
- $>=$: supérieur ou égal à
- $<=$: inférieur ou égal à

Les opérateurs logiques

- && : ET (donne 1 si les expressions sont simultanément réalisées)
- || : OU (inclusif : donne 1 si au moins une des expressions est réalisée)
- ! : NON (printf("%d",!(x==5)); donne 0 si x=5 et 1 sinon)

Questions-réponses parties 4

- Corrigez les erreurs :

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } $\longrightarrow$ for(i=0;i<=imax;i++) { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } $\longrightarrow$ for(i=0;i<=imax;i++) { ... }

  while(i=0;i<=imax;i++); { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } → for(i=0;i<=imax;i++) { ... }

  while(i=0;i<=imax;i++); { ... } → while(i<=imax) { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } $\longrightarrow$ for(i=0;i<=imax;i++) { ... }

  while(i=0;i<=imax;i++); { ... } $\longrightarrow$ while(i<=imax) { ... }

  if(i=0) { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } → for(i=0;i<=imax;i++) { ... }

  while(i=0;i<=imax;i++); { ... } → while(i<=imax) { ... }

  if(i=0) { ... } → if(i==0) { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } → for(i=0;i<=imax;i++) { ... }

  while(i=0;i<=imax;i++); { ... } → while(i<=imax) { ... }

  if(i=0) { ... } → if(i==0) { ... }

  if(i!=0); { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

for(i=0;i<=imax); { ... } → for(i=0;i<=imax;i++) { ... }

while(i=0;i<=imax;i++); { ... } → while(i<=imax) { ... }

if(i=0) { ... } → if(i==0) { ... }

if(i!=0); { ... } → if(i!=0) { ... }

Questions-réponses parties 4

- Corrigez les erreurs :

```
for(i=0;i<=imax); { ... }          → for(i=0;i<=imax;i++) { ... }
while(i=0;i<=imax;i++); { ... }    → while(i<=imax) { ... }
if(i=0) { ... }                    → if(i==0) { ... }
if(i!=0); { ... }                  → if(i!=0)  { ... }
```

- Que donne l'instruction suivante ?

```
i=5;
j=4;
if((i==5)&&(j>=10))
{    printf("test 1 ok\n");    }
if((i==5)||(j>=10))
{    printf("test 2 ok\n");    }
```

Questions-réponses parties 4

- Corrigez les erreurs :

  for(i=0;i<=imax); { ... } → for(i=0;i<=imax;i++) { ... }

  while(i=0;i<=imax;i++); { ... } → while(i<=imax) { ... }

  if(i=0) { ... } → if(i==0) { ... }

  if(i!=0); { ... } → if(i!=0) { ... }

- Que donne l'instruction suivante ?

```
i=5;
j=4;
if((i==5)&&(j>=10))
{    printf("test 1 ok\n");    }
if((i==5)||(j>=10))
{    printf("test 2 ok\n");    }
```

  → test 2 ok

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\rightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\longrightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

  $\longrightarrow$ D'où : $\frac{dv}{dt} = -\frac{v}{\tau}$ où $\tau = m/\alpha$

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\longrightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

  $\longrightarrow$ D'où : $\frac{dv}{dt} = -\frac{v}{\tau}$ où $\tau = m/\alpha$
- Intégration numérique : méthode d'Euler (pas précis)

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\longrightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

  $\longrightarrow$ D'où : $\frac{dv}{dt} = -\frac{v}{\tau}$ où $\tau = m/\alpha$
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dv}{dt} = \lim_{\delta \to 0} \frac{v(t+\delta)-v(t)}{\delta} \simeq \frac{v(t+\delta)-v(t)}{\delta}$ si $\delta << 1$

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\longrightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

  $\longrightarrow$ D'où : $\frac{dv}{dt} = -\frac{v}{\tau}$ où $\tau = m/\alpha$
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dv}{dt} = \lim_{\delta \longrightarrow 0} \frac{v(t+\delta)-v(t)}{\delta} \simeq \frac{v(t+\delta)-v(t)}{\delta}$ si $\delta << 1$

  $\longrightarrow$ D'où : $v(t+\delta) = v(t)(1-\delta/\tau)$

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\longrightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

  $\longrightarrow$ D'où : $\frac{dv}{dt} = -\frac{v}{\tau}$ où $\tau = m/\alpha$
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dv}{dt} = \lim_{\delta \to 0} \frac{v(t+\delta)-v(t)}{\delta} \simeq \frac{v(t+\delta)-v(t)}{\delta}$ si $\delta << 1$

  $\longrightarrow$ D'où : $v(t+\delta) = v(t)(1-\delta/\tau)$
- Pour calculer v à l'instant $t+\delta$ il faut connaître v à l'instant t

Enfin un peu de Physique ...

- Un bateau en panne de moteur se déplace en ligne droite à la vitesse v et est soumis à un frottement visqueux :
- Problème : comment évolue la vitesse du bateau ?

  $\longrightarrow$ Equation fondamentale de la dynamique : $\sum \mathcal{F} = ma = m\frac{dv}{dt} = -\alpha v$

  $\longrightarrow$ D'où : $\frac{dv}{dt} = -\frac{v}{\tau}$ où $\tau = m/\alpha$
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dv}{dt} = \lim_{\delta \to 0} \frac{v(t+\delta)-v(t)}{\delta} \simeq \frac{v(t+\delta)-v(t)}{\delta}$ si $\delta << 1$

  $\longrightarrow$ D'où : $v(t+\delta) = v(t)(1-\delta/\tau)$
- Pour calculer v à l'instant $t+\delta$ il faut connaître v à l'instant t

  $\longrightarrow$ Utiliser un tableau pour stocker v à l'instant t !

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

→ Nécessaire pour imprimer les résultats dans un fichier texte

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ tau : temps caractéristique de relaxation (fixe l'échelle de temps)

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ delta : pas de temps d'intégration (delta « 1)

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ Bornes min et sup du temps d'intégration

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ i : variable muette pour la boucle for

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ N : nombre total de points, N : nombre de cases du tableau v

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ t : temps; v[N] : tableau à N cases pour stocker la vitesse (de v[0] à v[N-1])

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ Initialisation de la vitesse et du temps

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

→ Boucle d'intégration de l'équation différentielle de la vitesse

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ A chaque pas, le temps s'écoule de delta

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ Calcul de la vitesse au temps t+delta connaissant la vitesse au temps t

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ La vitesse au temps t+delta est stockée dans la case i+1 du tableau v

programme_part5_1.c (dans main)

```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

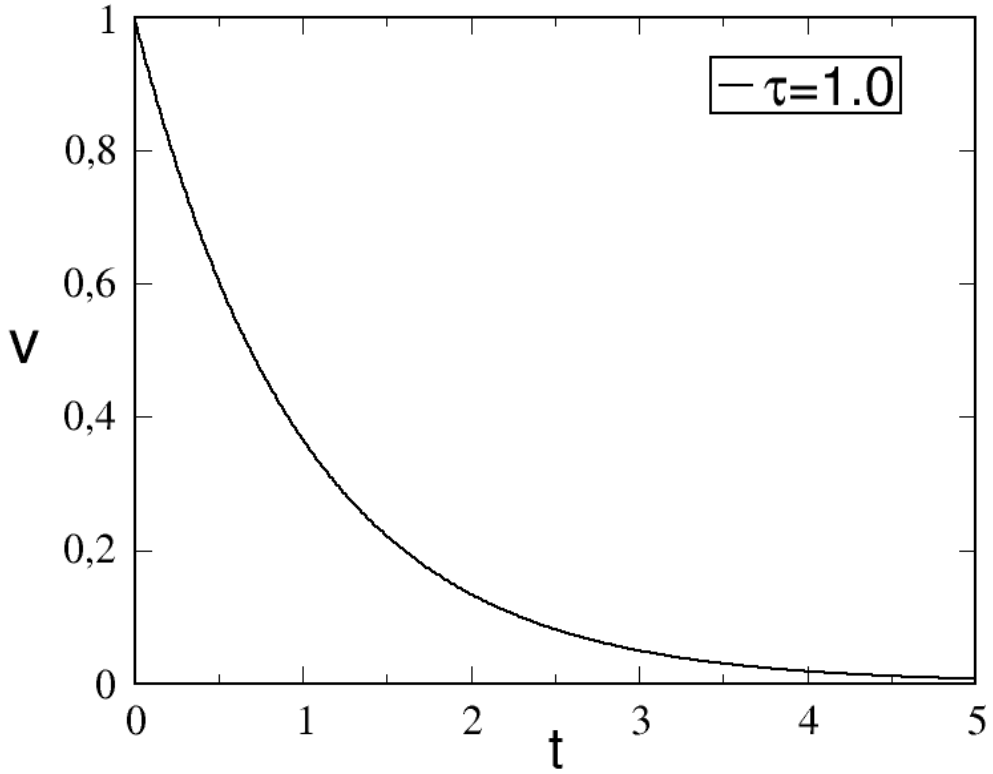
$\longrightarrow$ On imprime le temps et la vitesse correspondante dans le fichier out.txt

programme_part5_1.c (dans main)

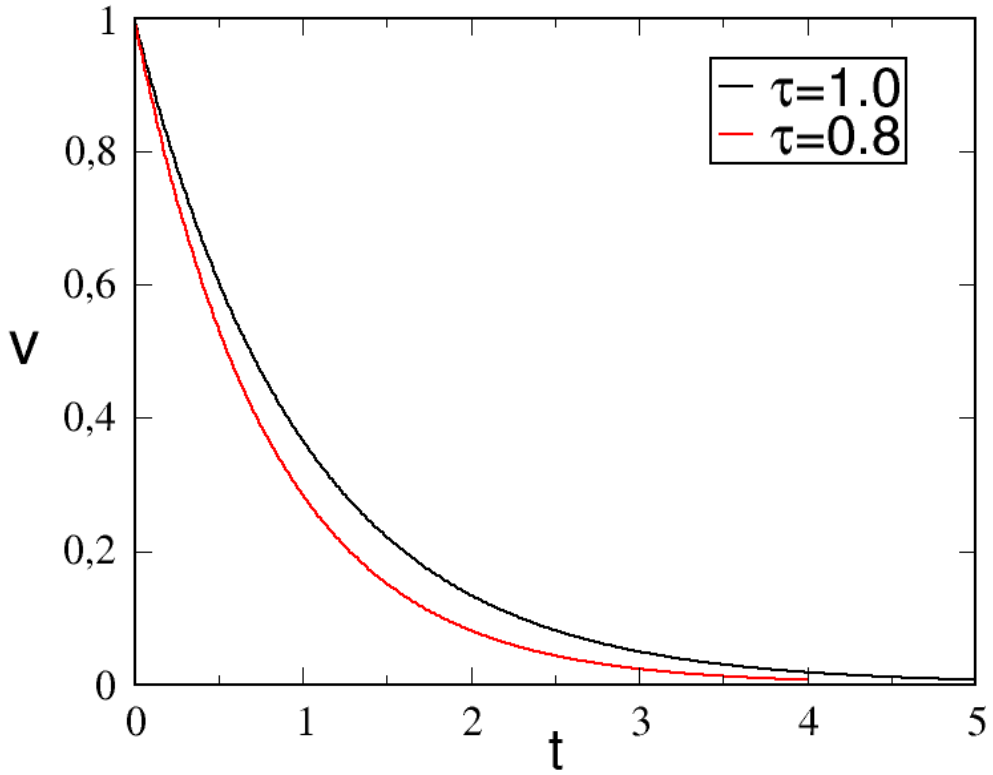
```
1:      FILE *file;
2:      file=fopen("out.txt","w");
3:
4:      double tau=1.0, delta=0.01;
5:      double a=0,b=5*tau;
6:      int i, N=(int)(b-a)/delta;
7:      double t, v[N];
8:
9:      v[0]=1.0;
10:     t=0.0;
11:
12:     for(i=0;i<N;i++)
13:     {
14:         t=t+delta;
15:         v[i+1]=v[i]*(1-delta/tau);
16:         fprintf(file,"%lf %lf\n",t,v[i+1]);
17:     }
18:     fclose(file);
```

$\longrightarrow$ On ferme le fichier out.txt

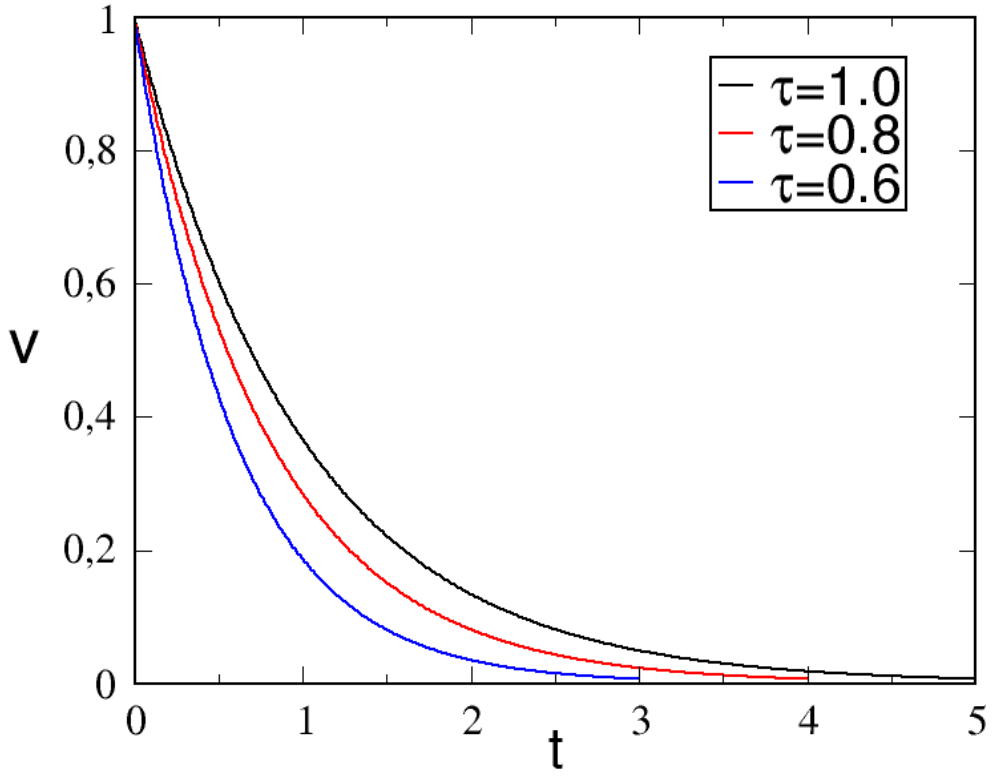
On obtient ...



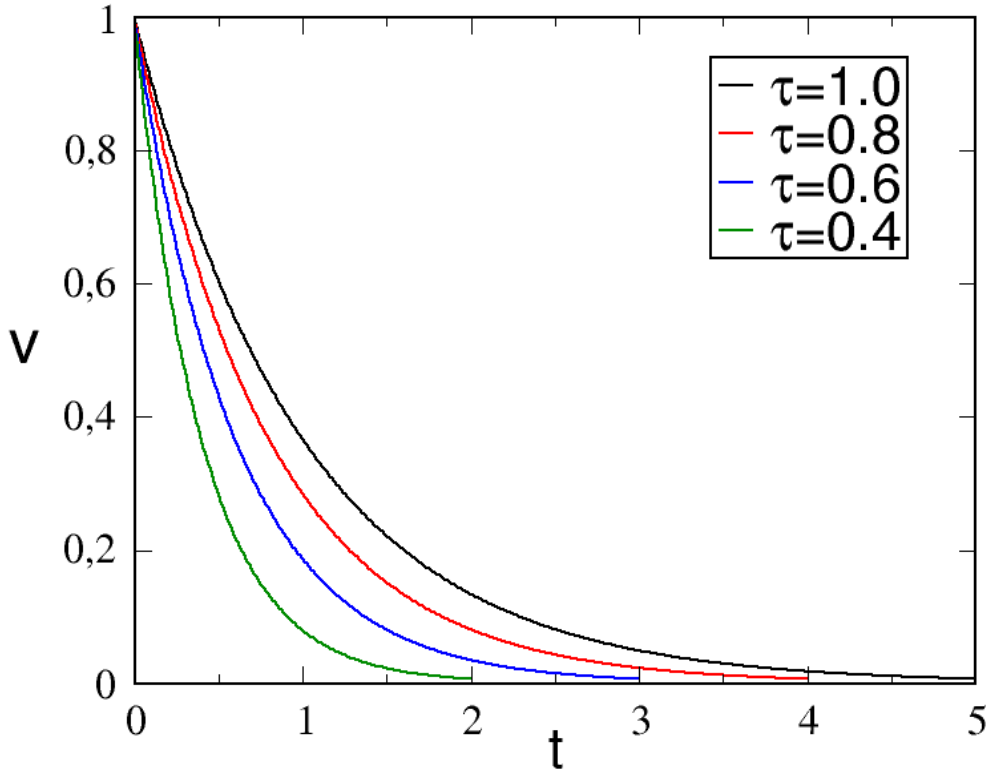
On obtient ...



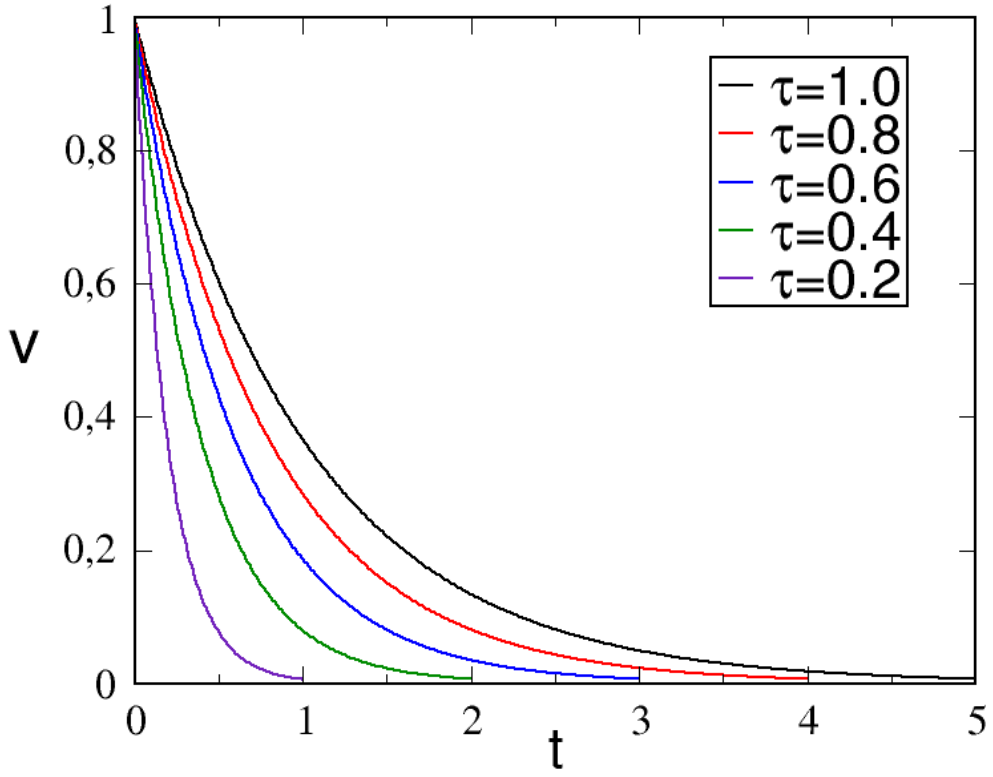
On obtient ...



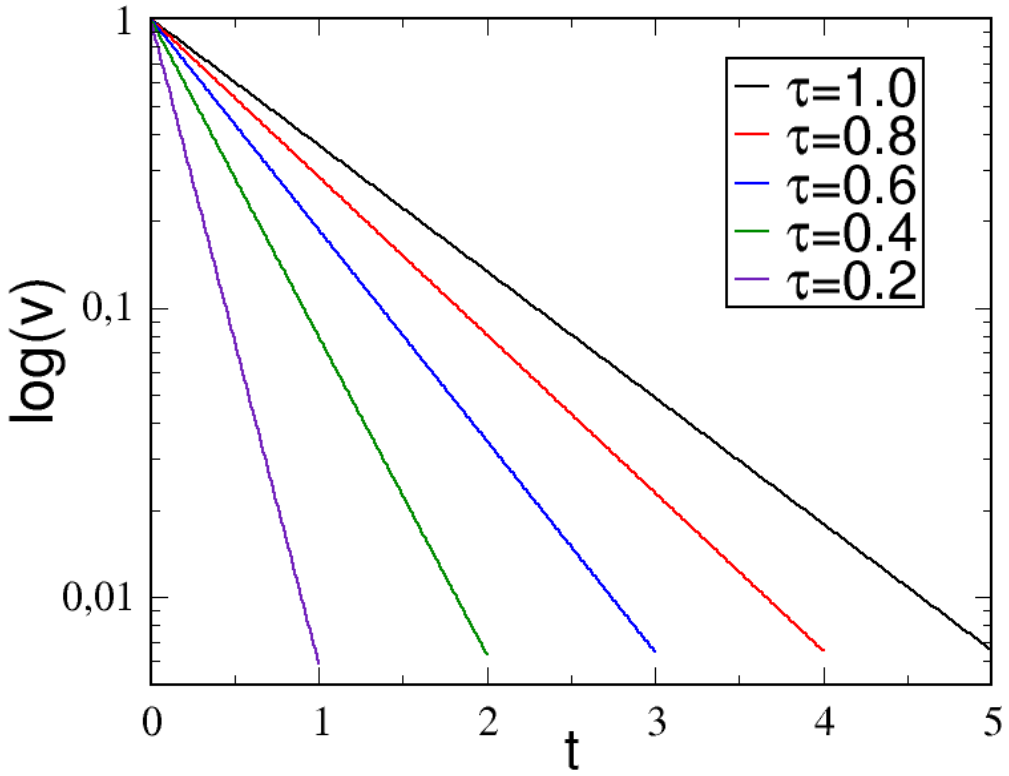
On obtient ...



On obtient ...



En échelle semi-log : droites $\longrightarrow$ exponentielles



Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

  $\longrightarrow \alpha$: tau de natalité; β : tau de mortalité par individu

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

  $\longrightarrow \alpha$: tau de natalité; β : tau de mortalité par individu
- Intégration numérique : méthode d'Euler (pas précis)

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

  $\longrightarrow \alpha$: tau de natalité; β : tau de mortalité par individu
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dn}{dt} = \lim_{\delta \to 0} \frac{n(t+\delta)-n(t)}{\delta} \simeq \frac{n(t+\delta)-n(t)}{\delta}$ si $\delta << 1$

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

  $\longrightarrow \alpha$: tau de natalité; β : tau de mortalité par individu
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dn}{dt} = \lim_{\delta \to 0} \frac{n(t+\delta)-n(t)}{\delta} \simeq \frac{n(t+\delta)-n(t)}{\delta}$ si $\delta << 1$

  $\longrightarrow$ D'où : $n(t+\delta) = n(t)(1+\alpha\delta - \beta\delta n(t))$

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

  $\longrightarrow \alpha$: tau de natalité; β : tau de mortalité par individu
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dn}{dt} = \lim_{\delta \to 0} \frac{n(t+\delta)-n(t)}{\delta} \simeq \frac{n(t+\delta)-n(t)}{\delta}$ si $\delta << 1$

  $\longrightarrow$ D'où : $n(t+\delta) = n(t)(1+\alpha\delta - \beta\delta n(t))$
- Pour calculer n à l'instant $t+\delta$ il faut connaître n à l'instant t

Exemple de dynamique de population ...

- Soit une population de coccinelle dans un espace clôt avec une nourriture abondante :
- Problème : comment évolue le nombre de coccinelle n(t) ?

  $\longrightarrow \frac{dn}{dt} = \alpha n - \beta n^2$

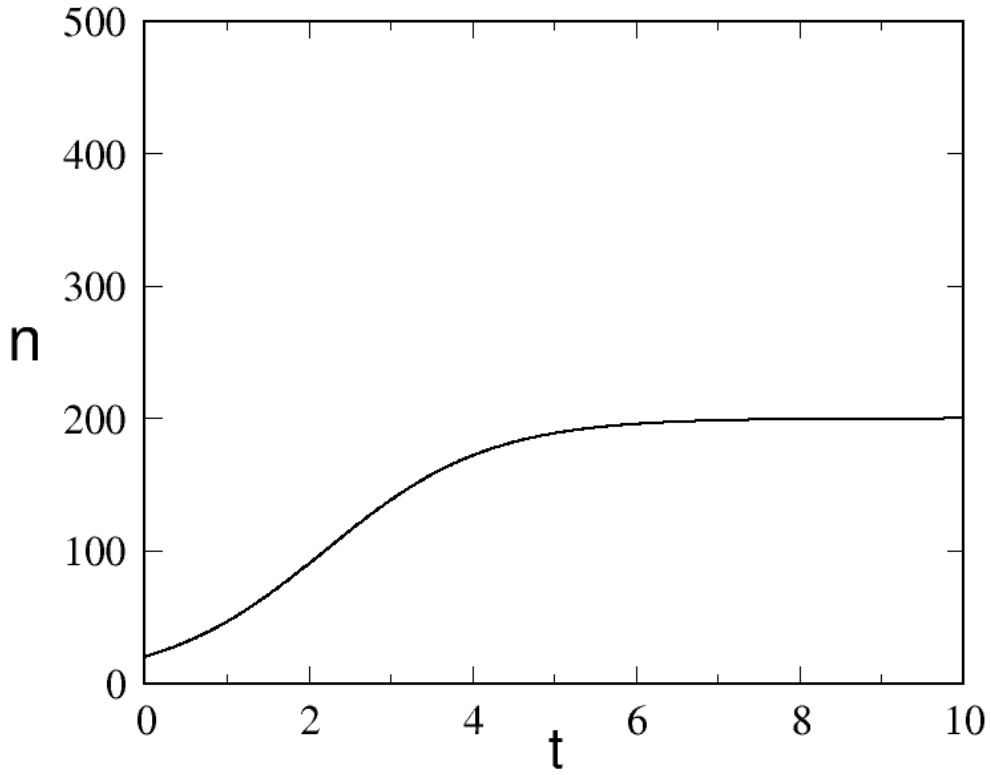
  $\longrightarrow \alpha$: tau de natalité; β : tau de mortalité par individu
- Intégration numérique : méthode d'Euler (pas précis)

  $\frac{dn}{dt} = \lim_{\delta \to 0} \frac{n(t+\delta)-n(t)}{\delta} \simeq \frac{n(t+\delta)-n(t)}{\delta}$ si $\delta << 1$

  $\longrightarrow$ D'où : $n(t+\delta) = n(t)(1+\alpha\delta - \beta\delta n(t))$
- Pour calculer n à l'instant $t+\delta$ il faut connaître n à l'instant t

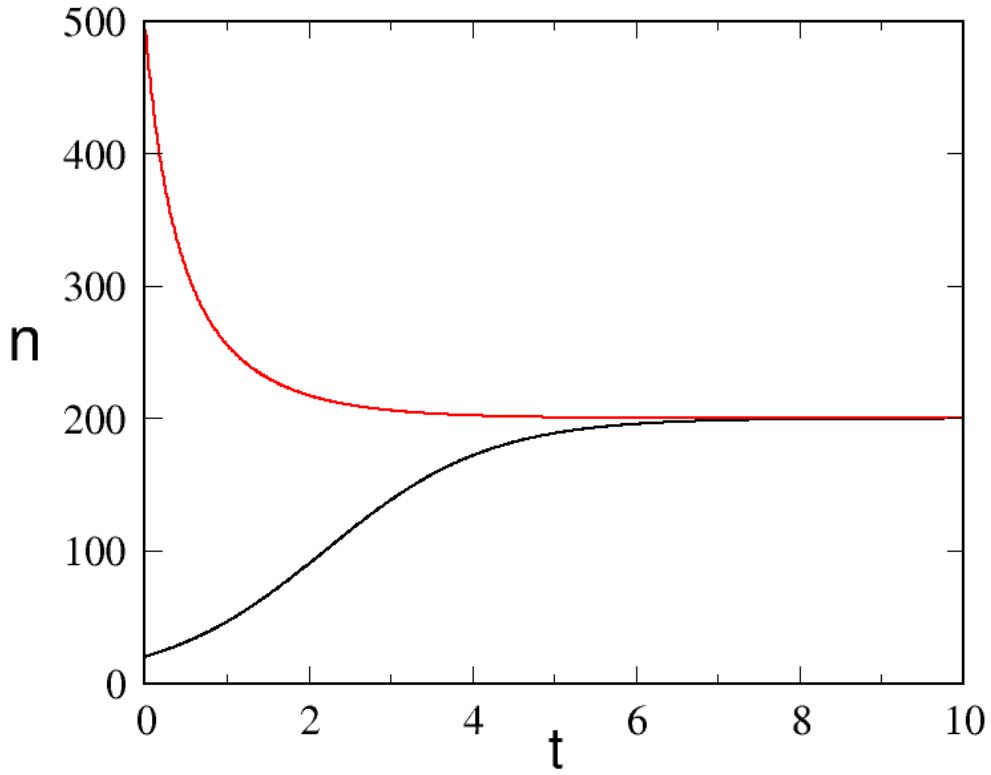
  $\longrightarrow$ Utiliser un tableau pour stocker n à l'instant t !

Voir programme : programme_part5_2.c



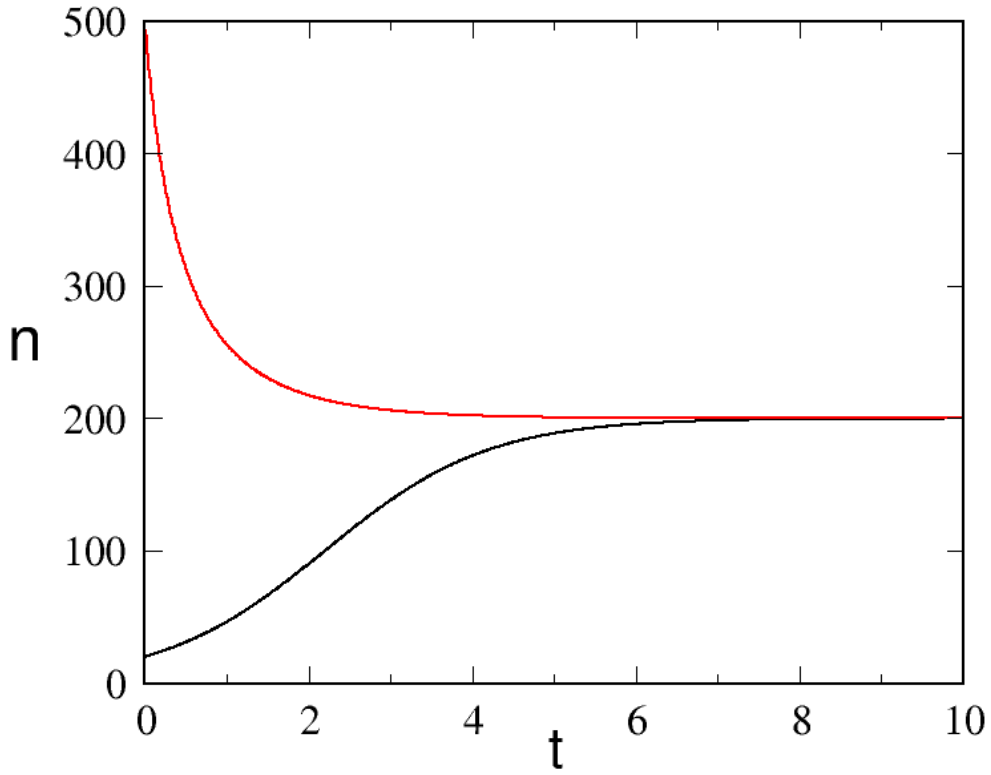
Population initiale : n(0)=20 coccinelles

Voir programme : programme_part5_2.c



Population initiale : n(0)=500 coccinelles

Voir programme : programme_part5_2.c



Le nombre de coccinelles converge vers $n_{eq} = \alpha/\beta = 200$

Utiliser une fonction C

- Qu'est-ce qu'une fonction ?

Utiliser une fonction C

- Qu'est-ce qu'une fonction ?

  $\longrightarrow$ portion de code qui peut travailler avec des données qu'on lui fournit

Utiliser une fonction C

- Qu'est-ce qu'une fonction ?

  $\longrightarrow$ portion de code qui peut travailler avec des données qu'on lui fournit

  $\longrightarrow$ elle peut renvoyer un résultat, elle peut être utilisée plusieurs fois

Utiliser une fonction C

- Qu'est-ce qu'une fonction ?

  $\longrightarrow$ portion de code qui peut travailler avec des données qu'on lui fournit

  $\longrightarrow$ elle peut renvoyer un résultat, elle peut être utilisée plusieurs fois

  $\longrightarrow$ elle permet de ne pas réécrire à chaque fois ce code

Utiliser une fonction C

- Qu'est-ce qu'une fonction ?

  $\longrightarrow$ portion de code qui peut travailler avec des données qu'on lui fournit

  $\longrightarrow$ elle peut renvoyer un résultat, elle peut être utilisée plusieurs fois

  $\longrightarrow$ elle permet de ne pas réécrire à chaque fois ce code

- Déclaration et utilisation facile

Utiliser une fonction C

- Qu'est-ce qu'une fonction ?

  $\longrightarrow$ portion de code qui peut travailler avec des données qu'on lui fournit

  $\longrightarrow$ elle peut renvoyer un résultat, elle peut être utilisée plusieurs fois

  $\longrightarrow$ elle permet de ne pas réécrire à chaque fois ce code

- Déclaration et utilisation facile

- Exemple de fonctions : main, printf et scanf (déclarée dans stdio.h)

programme_part5_3.c (problème du bateau bis)

```
1:  #include <stdio.h>
2:  # define tau 1.0
3:  # define delta 0.01
4:
5:  double f(double v)
6:  {
7:      return v*(1-delta/tau);
8:  }
9:  int main()
10: {
11:     double a=0,b=5*tau;
12:     int i, N=(int)(b-a)/delta;
13:     double t, v[N];
14:
15:     v[0]=1.0;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         t=t+delta;
21:         v[i+1]=f(v[i]);
22:         printf("%lf %lf\n",t,v[i+1]);
23:     }
24: return 0;
25: }
```

→ Librairies et déclarations globales des constantes tau et delta

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ On déclare une fonction nommée f qui va utiliser un nombre décimale v

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ La fonction f renverra en sortie le nombre v*(1-delta/tau)

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ Boucle principale "main"

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

→ On déclare les bornes, le nombre de pas, le temps et le tableau vitesse v[N]

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ On initialise les variables vitesse et temps

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ Boucle qui calcule la vitesse du bateau via la fonction f

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ Le temps s'écoule de delta à chaque pas

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ On calcule la valeur de la vitesse au temps t+delta via la fonction f

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

→ On donne le nombre v[i] comme argument à la fonction f

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ La fonction f va comprendre que v=v[i] ...

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ ... et va calculer v*(1-delta/tau)

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ ... et renvoyer cette valeur : f(v[i])=v*(1-delta/tau)

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ on stocke ensuite cette valeur dans la case i+1 du tableau v ...

programme_part5_3.c (problème du bateau bis)

```
5:      double f(double v)
6:      {
7:          return v*(1-delta/tau);
8:      }
9:      int main()
10:     {
11:         double a=0,b=5*tau;
12:         int i, N=(int)(b-a)/delta;
13:         double t, v[N];
14:
15:         v[0]=1.0;
16:         t=0.0;
17:
18:         for(i=0;i<N;i++)
19:         {
20:             t=t+delta;
21:             v[i+1]=f(v[i]);
22:             printf("%lf %lf\n",t,v[i+1]);
23:         }
24:     return 0;
25:     }
```

$\longrightarrow$ Puis on imprime le temps t et la vitesse dans la console

En TDs/Tps sur ordinateur

- On utilisera souvent des fonctions pour résoudre des équations différentielles

En TDs/Tps sur ordinateur

- On utilisera souvent des fonctions pour résoudre des équations différentielles
- On utilisera une méthode beaucoup plus précise : Runge Kutta 4 (RK4)

En TDs/Tps sur ordinateur

- On utilisera souvent des fonctions pour résoudre des équations différentielles
- On utilisera une méthode beaucoup plus précise : Runge Kutta 4 (RK4)
- On résoudra aussi des éqs. diffs. du second ordre (pendule ...)

En TDs/Tps sur ordinateur

- On utilisera souvent des fonctions pour résoudre des équations différentielles
- On utilisera une méthode beaucoup plus précise : Runge Kutta 4 (RK4)
- On résoudra aussi des éqs. diffs. du second ordre (pendule ...)
- Tout le contenu de ce cours sera utile !

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.

  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.

  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.
  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

  $\longrightarrow$ On peut définir un tableau à plusieurs dimensions. Oui, T[3][3] peut stocker une matrice 3*3

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.
  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

  $\longrightarrow$ On peut définir un tableau à plusieurs dimensions. Oui, T[3][3] peut stocker une matrice 3*3

- Corrigez les erreurs :

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.
  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

  $\longrightarrow$ On peut définir un tableau à plusieurs dimensions. Oui, T[3][3] peut stocker une matrice 3*3

- Corrigez les erreurs :

  double f(x) { return x; }

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.
  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

  $\longrightarrow$ On peut définir un tableau à plusieurs dimensions. Oui, T[3][3] peut stocker une matrice 3*3

- Corrigez les erreurs :

  double f(x) { return x; } $\longrightarrow$ double f(double x) { return x; }

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.
  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

  $\longrightarrow$ On peut définir un tableau à plusieurs dimensions. Oui, T[3][3] peut stocker une matrice 3*3

- Corrigez les erreurs :

  double f(x) { return x; } $\longrightarrow$ double f(double x) { return x; }

  double f(double x, double y) { return x*y; }

  v=f(x);

Questions-réponses parties 5

- Qu'est ce qu'un tableau ? Combien de cases possède le tableau T[10] ?

  $\longrightarrow$ Un tableau est un ensemble de variables de même type. Ces variables sont des éléments du tableau.
  $\longrightarrow$ on stocke leurs valeurs dans les cases i du tableau T[i]. Le tableau T[10] possède 10 cases (de T[0] à T[9] !)

- Puis-je utiliser un tableau 2D T[3][3] pour stocker une matrice 3*3 ?

  $\longrightarrow$ On peut définir un tableau à plusieurs dimensions. Oui, T[3][3] peut stocker une matrice 3*3

- Corrigez les erreurs :

  double f(x) { return x; } $\longrightarrow$ double f(double x) { return x; }

  double f(double x, double y) { return x*y; }

  v=f(x); $\longrightarrow$ v=f(x,y);

Deux problèmes de dynamique

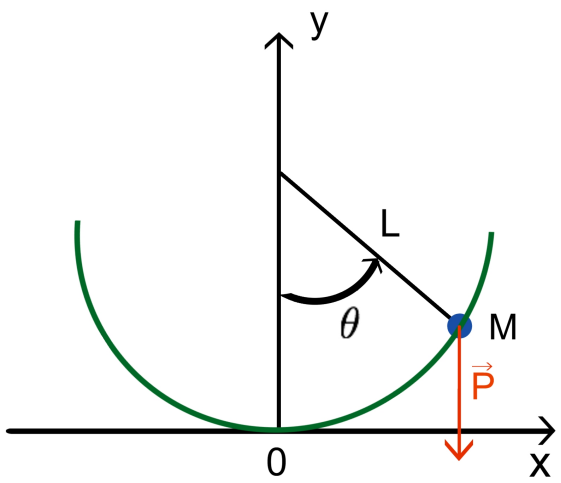
- Pendule pesant (exo de TDs/TPs)
- Deux corps en interaction gravitationnelle

Pendule pesant

- Energie :

$\mathcal{E} = \mathcal{E}_c + \mathcal{E}_p = cste$ (forces conservatives)

$\rightarrow \mathcal{E} = \frac{1}{2} M(L\dot{\theta})^2 + MgL(1 - \cos(\theta))$



- Equation du mouvement :

$\rightarrow \ddot{\theta} + \omega^2 \sin(\theta) = 0$ avec $\omega^2 = \frac{g}{L}$

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  Rappel : $\sin(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{(2n+1)}}{(2n+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots$

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  Rappel : $\sin(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{(2n+1)}}{(2n+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots$

- Petits angles ($\theta < 30^\circ$) : $\sin(\theta) \simeq \theta$ d'où $\ddot{\theta} + \omega^2 \theta = 0$

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  Rappel : $\sin(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{(2n+1)}}{(2n+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots$

- Petits angles ($\theta < 30^\circ$) : $\sin(\theta) \simeq \theta$ d'où $\ddot{\theta} + \omega^2 \theta = 0$

  $\longrightarrow$ Solutions générales : $\theta(t) = A\cos(\omega t) + B\sin(\omega t)$

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  Rappel : $\sin(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{(2n+1)}}{(2n+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots$

- Petits angles ($\theta < 30\,^\circ$) : $\sin(\theta) \simeq \theta$ d'où $\ddot{\theta} + \omega^2 \theta = 0$

  $\longrightarrow$ Solutions générales : $\theta(t) = A\cos(\omega t) + B\sin(\omega t)$

- Petits et grands angles : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  Rappel : $\sin(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{(2n+1)}}{(2n+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots$

- Petits angles ($\theta < 30\,^\circ$) : $\sin(\theta) \simeq \theta$ d'où $\ddot{\theta} + \omega^2 \theta = 0$

  $\longrightarrow$ Solutions générales : $\theta(t) = A\cos(\omega t) + B\sin(\omega t)$

- Petits et grands angles : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  $\longrightarrow$ Solutions générales : fonctions elliptiques de Jacobi (cnoïdales)

Pendule pesant

- Eq. du mouvement : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  Rappel : $\sin(\theta) = \sum_{n=0}^{\infty} \frac{(-1)^n \theta^{(2n+1)}}{(2n+1)!} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} + \ldots$

- Petits angles ($\theta < 30\,^\circ$) : $\sin(\theta) \simeq \theta$ d'où $\ddot{\theta} + \omega^2 \theta = 0$

  $\longrightarrow$ Solutions générales : $\theta(t) = A\cos(\omega t) + B\sin(\omega t)$

- Petits et grands angles : $\ddot{\theta} + \omega^2 \sin(\theta) = 0$

  $\longrightarrow$ Solutions générales : fonctions elliptiques de Jacobi (cnoïdales)

- Comment résoudre numériquement cette équation différentielle $\forall\ \theta$?

Pendule pesant

- On connaît $\theta(0)$, donc $\ddot{\theta}(0)$, et $\dot{\theta}(0)$ à t=0

Pendule pesant

- On connaît $\theta(0)$, donc $\ddot{\theta}(0)$, et $\dot{\theta}(0)$ à t=0
- On intègre $\ddot{\theta} \rightarrow$ on obtient $\dot{\theta}$

Pendule pesant

- On connaît $\theta(0)$, donc $\ddot{\theta}(0)$, et $\dot{\theta}(0)$ à t=0
- On intègre $\ddot{\theta} \longrightarrow$ on obtient $\dot{\theta}$
- On intègre maintenant $\dot{\theta} \longrightarrow$ on obtient θ

Pendule pesant

- On connaît $\theta(0)$, donc $\ddot{\theta}(0)$, et $\dot{\theta}(0)$ à t=0
- On intègre $\ddot{\theta} \longrightarrow$ on obtient $\dot{\theta}$
- On intègre maintenant $\dot{\theta} \longrightarrow$ on obtient θ
- Comment intégrer ?

  $\longrightarrow$Deux méthodes : Euler (pas bien) ou RK4 (bien)

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

$\longrightarrow$ Librairie math.h pour sin() et M_PI et définition de la pulsation $\omega = 1$

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

$\longrightarrow$ Fonction g qui prend θ (theta) et renvoie la valeur $-\omega^2 \sin(\theta)$

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

$\longrightarrow$ Fonction f qui prend $\dot{\theta}$ (thetapoint) et renvoie la même valeur $\dot{\theta}$!

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:   double g(double theta) { return -w*w*sin(theta); }
6:   double f(double thetapoint) { return thetapoint; }
7:   int main()
8:   {
9:         FILE *file;
10:        file=fopen("pendule.dat","w");
11:        double theta, thetapoint, t, dt=0.01;
12:        int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:        theta=0;
15:        thetapoint=1.9999;
16:        t=0.0;
17:
18:        for(i=0;i<N;i++)
19:        {
20:              thetapoint=thetapoint+dt*g(theta);
21:              theta=theta+dt*f(thetapoint);
22:              t=t+dt;
23:              fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:        }
25:  fclose(file);
26:  return 0;
27:  }
```

→ On va écrire dans le fichier texte "pendule.dat"

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

$\longrightarrow$ theta= angle, thetapoint= vitesse angulaire, t= temps, dt=pas de temps

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:   double g(double theta) { return -w*w*sin(theta); }
6:   double f(double thetapoint) { return thetapoint; }
7:   int main()
8:   {
9:         FILE *file;
10:        file=fopen("pendule.dat","w");
11:        double theta, thetapoint, t, dt=0.01;
12:        int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:        theta=0;
15:        thetapoint=1.9999;
16:        t=0.0;
17:
18:        for(i=0;i<N;i++)
19:        {
20:              thetapoint=thetapoint+dt*g(theta);
21:              theta=theta+dt*f(thetapoint);
22:              t=t+dt;
23:              fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:        }
25:  fclose(file);
26:  return 0;
27:  }
```

$\longrightarrow$ N=nombre total de points (itérations) = 10*T/dt ou T=$2\pi/\omega$

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:   double g(double theta) { return -w*w*sin(theta); }
6:   double f(double thetapoint) { return thetapoint; }
7:   int main()
8:   {
9:         FILE *file;
10:        file=fopen("pendule.dat","w");
11:        double theta, thetapoint, t, dt=0.01;
12:        int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:        theta=0;
15:        thetapoint=1.9999;
16:        t=0.0;
17:
18:        for(i=0;i<N;i++)
19:        {
20:              thetapoint=thetapoint+dt*g(theta);
21:              theta=theta+dt*f(thetapoint);
22:              t=t+dt;
23:              fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:        }
25:  fclose(file);
26:  return 0;
27:  }
```

→ Initialisation des variables : angle, vitesse et temps

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:   double g(double theta) { return -w*w*sin(theta); }
6:   double f(double thetapoint) { return thetapoint; }
7:   int main()
8:   {
9:         FILE *file;
10:        file=fopen("pendule.dat","w");
11:        double theta, thetapoint, t, dt=0.01;
12:        int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:        theta=0;
15:        thetapoint=1.9999;
16:        t=0.0;
17:
18:        for(i=0;i<N;i++)
19:        {
20:              thetapoint=thetapoint+dt*g(theta);
21:              theta=theta+dt*f(thetapoint);
22:              t=t+dt;
23:              fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:        }
25:  fclose(file);
26:  return 0;
27:  }
```

$\longrightarrow$ Vitesse initiale maximum pour osciller: $\dot{\theta} = 2$ (POURQUOI ?)

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

$\longrightarrow$ On va intégrer étape par étape, stop quand i=N-1

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:   double g(double theta) { return -w*w*sin(theta); }
6:   double f(double thetapoint) { return thetapoint; }
7:   int main()
8:   {
9:         FILE *file;
10:        file=fopen("pendule.dat","w");
11:        double theta, thetapoint, t, dt=0.01;
12:        int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:        theta=0;
15:        thetapoint=1.9999;
16:        t=0.0;
17:
18:        for(i=0;i<N;i++)
19:        {
20:              thetapoint=thetapoint+dt*g(theta);
21:              theta=theta+dt*f(thetapoint);
22:              t=t+dt;
23:              fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:        }
25:  fclose(file);
26:  return 0;
27:  }
```

→ On calcule la nouvelle vitesse en intégrant l'accélération via g()

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

$\longrightarrow$ On calcule le nouvel angle en intégrant la vitesse via f()

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:  double g(double theta) { return -w*w*sin(theta); }
6:  double f(double thetapoint) { return thetapoint; }
7:  int main()
8:  {
9:      FILE *file;
10:     file=fopen("pendule.dat","w");
11:     double theta, thetapoint, t, dt=0.01;
12:     int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:     theta=0;
15:     thetapoint=1.9999;
16:     t=0.0;
17:
18:     for(i=0;i<N;i++)
19:     {
20:         thetapoint=thetapoint+dt*g(theta);
21:         theta=theta+dt*f(thetapoint);
22:         t=t+dt;
23:         fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:     }
25: fclose(file);
26: return 0;
27: }
```

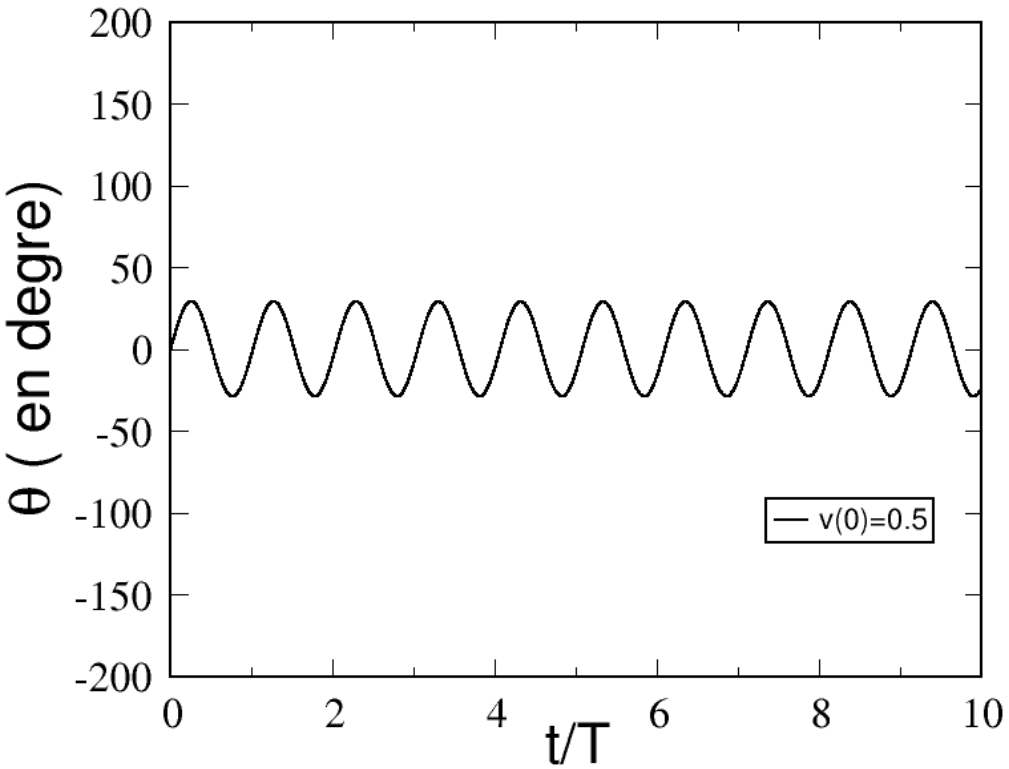
→ Le temps s'écoule de dt à chaque itération

Avec la méthode d'Euler

```
1:  #include <stdio.h>
2:  #include <math.h>
3:  # define w 1
4:
5:   double g(double theta) { return -w*w*sin(theta); }
6:   double f(double thetapoint) { return thetapoint; }
7:   int main()
8:   {
9:         FILE *file;
10:        file=fopen("pendule.dat","w");
11:        double theta, thetapoint, t, dt=0.01;
12:        int i, N=(int)((10.0*2*M_PI)/(w*dt));
13:
14:        theta=0;
15:        thetapoint=1.9999;
16:        t=0.0;
17:
18:        for(i=0;i<N;i++)
19:        {
20:              thetapoint=thetapoint+dt*g(theta);
21:              theta=theta+dt*f(thetapoint);
22:              t=t+dt;
23:              fprintf(file,"%lf     %lf\n", t, theta*180/M_PI);
24:        }
25:  fclose(file);
26:  return 0;
27:  }
```

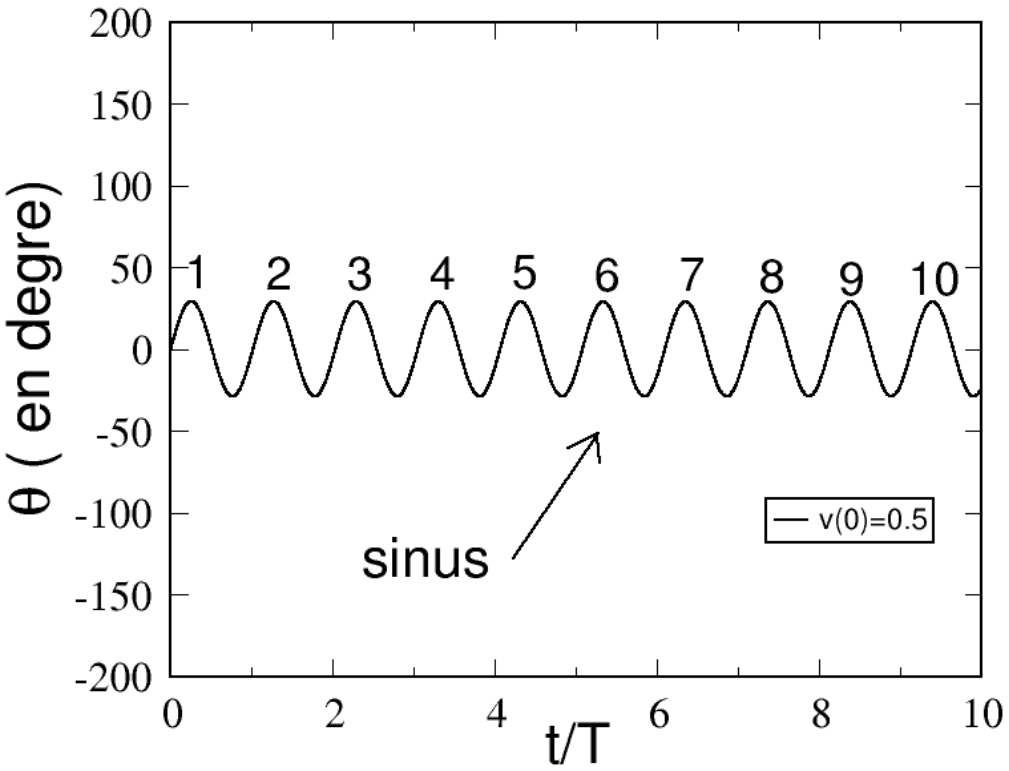
→ On imprime dans le fichier de sortie le temps et l'angle (en degré)

Pendule pesant



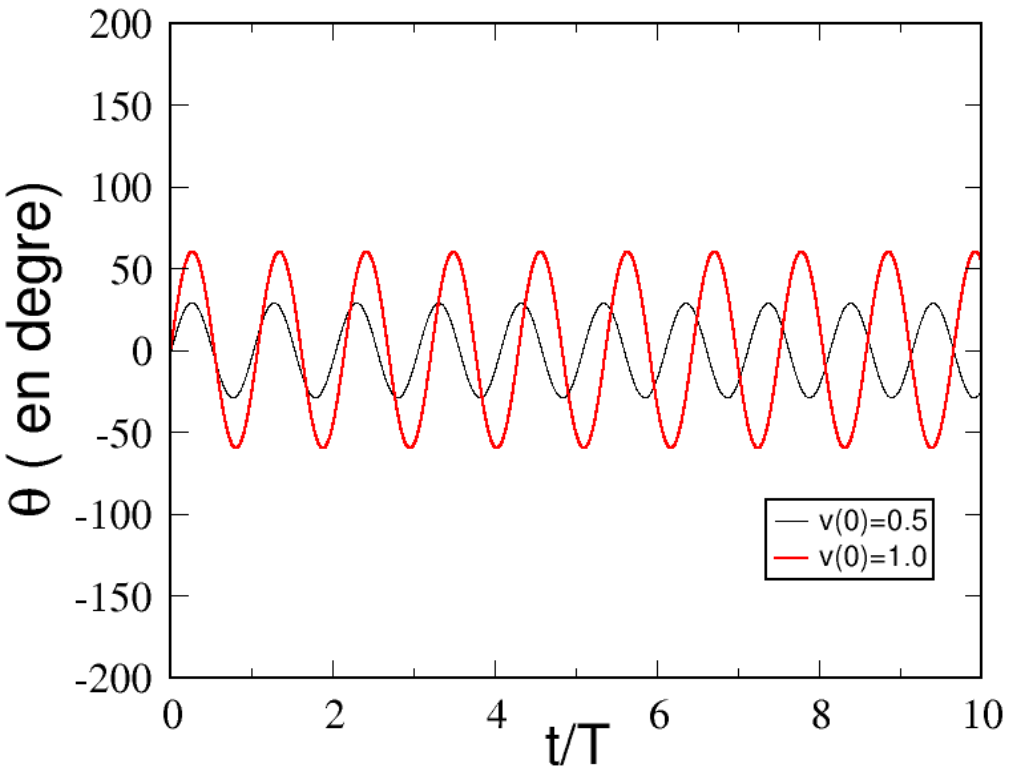
Petites oscillations : sinusoïde

Pendule pesant



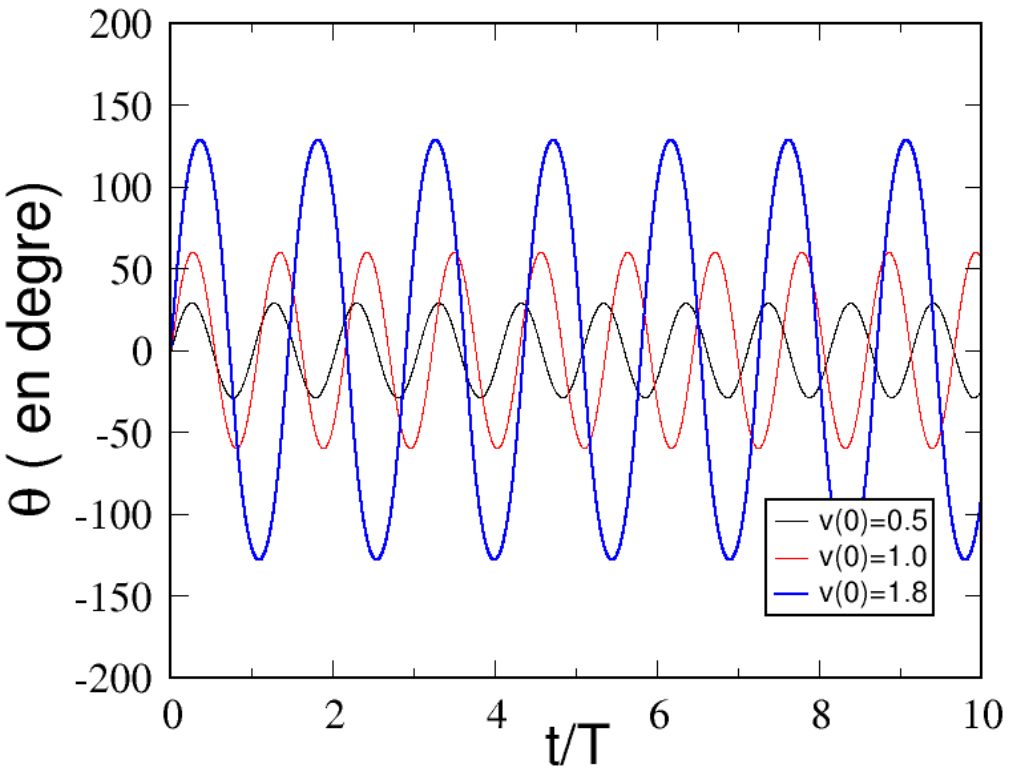
La période des oscillations est très proche de $T = 2\pi\sqrt{L/g}$

Pendule pesant



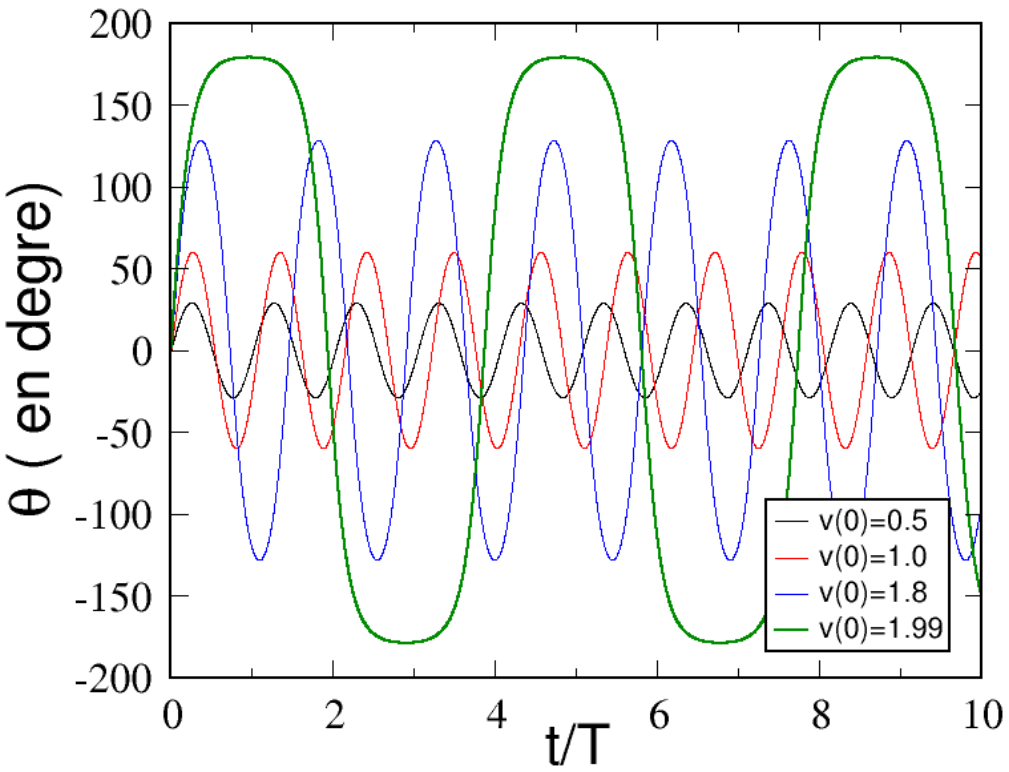
On augmente la vitesse initiale : la période augmente

Pendule pesant



On augmente la vitesse initiale : la période augmente

Pendule pesant



Grandes oscillations : fonctions elliptiques de Jacobi

Avec la méthode de Runge-Kutta d'ordre 4 (RK4)

- Problème quelconque : $y' = f(t, y)$ avec $y(t_0) = y_0$

Avec la méthode de Runge-Kutta d'ordre 4 (RK4)

- Problème quelconque : $y' = f(t, y)$ avec $y(t_0) = y_0$
- Au temps t+δ, on a $y(t+\delta) = y(t) + \frac{\delta}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Avec la méthode de Runge-Kutta d'ordre 4 (RK4)

- Problème quelconque : $y' = f(t, y)$ avec $y(t_0) = y_0$
- Au temps t+δ, on a $y(t+\delta) = y(t) + \frac{\delta}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

  Avec :

  $k_1 = f(t, y)$
  $k_2 = f(t + \frac{\delta}{2}, y + \frac{\delta}{2} k_1)$
  $k_3 = f(t + \frac{\delta}{2}, y + \frac{\delta}{2} k_2)$
  $k_4 = f(t + \delta, y + \delta\, k_3)$

Avec la méthode de Runge-Kutta d'ordre 4 (RK4)

- Problème quelconque : $y' = f(t, y)$ avec $y(t_0) = y_0$
- Au temps t+δ, on a $y(t+\delta) = y(t) + \frac{\delta}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

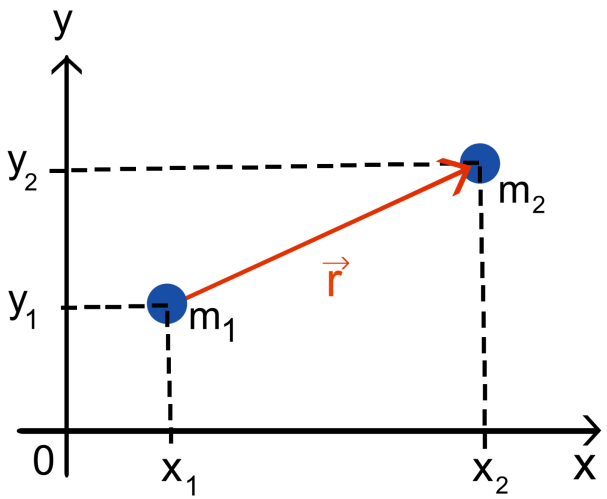
  Avec :

  $k_1 = f(t, y)$
  $k_2 = f(t + \frac{\delta}{2}, y + \frac{\delta}{2} k_1)$
  $k_3 = f(t + \frac{\delta}{2}, y + \frac{\delta}{2} k_2)$
  $k_4 = f(t + \delta, y + \delta\, k_3)$
- Comparaison avec Euler : $y(t+\delta) = y(t) + \delta\, k_1$

Deux corps en interaction gravitationnelle

- Forces :

$$\vec{F}_{2\to 1} = m_1 \vec{a}_1 = \frac{G m_1 m_2}{r^3}\vec{r}$$
$$\vec{F}_{1\to 2} = m_2 \vec{a}_2 = -\frac{G m_1 m_2}{r^3}\vec{r} = -\vec{F}_{2\to 1}$$
$$\text{Avec } \vec{r} = (x_2 - x_1)\hat{x} + (y_2 - y_1)\hat{y}$$
$$\text{d'où } r = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



- Equations du mouvement :

$$\ddot{x_1} = \frac{G m_2}{r^3}(x_2 - x_1)$$
$$\ddot{y_1} = \frac{G m_2}{r^3}(y_2 - y_1)$$
$$\ddot{x_2} = -\frac{G m_1}{r^3}(x_2 - x_1)$$
$$\ddot{y_2} = -\frac{G m_1}{r^3}(y_2 - y_1)$$

Deux corps en interaction gravitationnelle

- Avec la méthode d'Euler, on va intégrer $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$

Deux corps en interaction gravitationnelle

- Avec la méthode d'Euler, on va intégrer $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$
- On obtient donc les vitesses : $\dot{x_1}, \dot{y_1}, \dot{x_2}$ et $\dot{y_2}$

Deux corps en interaction gravitationnelle

- Avec la méthode d'Euler, on va intégrer $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$
- On obtient donc les vitesses : $\dot{x_1}, \dot{y_1}, \dot{x_2}$ et $\dot{y_2}$
- ... qu'on intègre pour avoir les positions : $x_1(t), y_1(t), x_2(t)$ et $y_2(t)$

Deux corps en interaction gravitationnelle

- Avec la méthode d'Euler, on va intégrer $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$
- On obtient donc les vitesses : $\dot{x_1}, \dot{y_1}, \dot{x_2}$ et $\dot{y_2}$
- ... qu'on intègre pour avoir les positions : $x_1(t), y_1(t), x_2(t)$ et $y_2(t)$
- Exemple avec la position x_1 :

$$\dot{x_1}(t) = \dot{x_1}(t-\delta) + \delta \ddot{x_1}(t-\delta) \text{ avec } \delta << 1$$

$$x_1(t+\delta) = x_1(t) + \delta \dot{x_1}(t)$$

Deux corps en interaction gravitationnelle

- Vectoriellement : $\vec{r}(t) = (x_1(t), y_1(t), x_2(t), y_2(t))$

Deux corps en interaction gravitationnelle

- Vectoriellement : $\vec{r}(t) = (x_1(t), y_1(t), x_2(t), y_2(t))$
- A t=0 : $\dot{\vec{r}}(0)$ et $\vec{r}(0)$ donc $\ddot{\vec{r}}(0)$ car $\ddot{\vec{r}}(0) = f(\vec{r}(0))$

Deux corps en interaction gravitationnelle

- Vectoriellement : $\vec{r}(t) = (x_1(t), y_1(t), x_2(t), y_2(t))$
- A t=0 : $\dot{\vec{r}}(0)$ et $\vec{r}(0)$ donc $\ddot{\vec{r}}(0)$ car $\ddot{\vec{r}}(0) = f(\vec{r}(0))$
- A t=δ :

  $\dot{\vec{r}}(\delta) = \dot{\vec{r}}(0) + \delta\ddot{\vec{r}}(0)$

  $\vec{r}(\delta) = \vec{r}(0) + \delta\dot{\vec{r}}(0)$ d'où $\ddot{\vec{r}}(\delta)$

Deux corps en interaction gravitationnelle

- Vectoriellement : $\vec{r}(t) = (x_1(t), y_1(t), x_2(t), y_2(t))$
- A t=0 : $\dot{\vec{r}}(0)$ et $\vec{r}(0)$ donc $\ddot{\vec{r}}(0)$ car $\ddot{\vec{r}}(0) = f(\vec{r}(0))$
- A t=δ :

  $\dot{\vec{r}}(\delta) = \dot{\vec{r}}(0) + \delta\ddot{\vec{r}}(0)$

  $\vec{r}(\delta) = \vec{r}(0) + \delta\dot{\vec{r}}(0)$ d'où $\ddot{\vec{r}}(\delta)$
- A t=2δ :

  $\dot{\vec{r}}(2\delta) = \dot{\vec{r}}(\delta) + \delta\ddot{\vec{r}}(\delta)$

  $\vec{r}(2\delta) = \vec{r}(\delta) + \delta\dot{\vec{r}}(\delta)$ d'où $\ddot{\vec{r}}(2\delta)$

Deux corps en interaction gravitationnelle

- Vectoriellement : $\vec{r}(t) = (x_1(t), y_1(t), x_2(t), y_2(t))$
- A t=0 : $\dot{\vec{r}}(0)$ et $\vec{r}(0)$ donc $\ddot{\vec{r}}(0)$ car $\ddot{\vec{r}}(0) = f(\vec{r}(0))$
- A t=δ :

  $\dot{\vec{r}}(\delta) = \dot{\vec{r}}(0) + \delta\ddot{\vec{r}}(0)$

  $\vec{r}(\delta) = \vec{r}(0) + \delta\dot{\vec{r}}(0)$ d'où $\ddot{\vec{r}}(\delta)$
- A t=2δ :

  $\dot{\vec{r}}(2\delta) = \dot{\vec{r}}(\delta) + \delta\ddot{\vec{r}}(\delta)$

  $\vec{r}(2\delta) = \vec{r}(\delta) + \delta\dot{\vec{r}}(\delta)$ d'où $\ddot{\vec{r}}(2\delta)$
- etc ...

Deux corps en interaction gravitationnelle

- Dans le programme : 4 fonctions fx1, fy1, fx2 et fy2 qui renvoient les accélérations $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$

Deux corps en interaction gravitationnelle

- Dans le programme : 4 fonctions fx1, fy1, fx2 et fy2 qui renvoient les accélérations $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$
- Ces 4 fonctions ont comme arguments les positions x_1, y_1, x_2 et y_2

Deux corps en interaction gravitationnelle

- Dans le programme : 4 fonctions fx1, fy1, fx2 et fy2 qui renvoient les accélérations $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$
- Ces 4 fonctions ont comme arguments les positions x_1, y_1, x_2 et y_2
- On utilisera aussi une autre fonction g qui prend la vitesse et renvoie la vitesse

Deux corps en interaction gravitationnelle

- Dans le programme : 4 fonctions fx1, fy1, fx2 et fy2 qui renvoient les accélérations $\ddot{x_1}, \ddot{y_1}, \ddot{x_2}$ et $\ddot{y_2}$
- Ces 4 fonctions ont comme arguments les positions x_1, y_1, x_2 et y_2
- On utilisera aussi une autre fonction g qui prend la vitesse et renvoie la vitesse
- Effectuez des "tests" simples pour sonder la validité de votre programme !

Programmer en langage C :

Un outil pour physicien – Introduction

Programmes du Cours de Licence 3 Physique
Université de Nice Sophia-Antipolis
Nice 2010-2011

Laurent de FORGES de PARNY
email: de.forges.de.parny.laurent@etu.unice.fr

Université Nice SOPHIA ANTIPOLIS

Partie 1

---

programme_part1_1.c

```c
// Premier exemple de programme simple
#include <stdio.h>

int main()
{
    printf("vive la physique\n");
    return 0;
}
```

programme_part1_2.c

```c
/* Premier exemple de programme simple */
#include <stdio.h>

int main()
{
    printf("vive la physique\n");
    return 0;
}
```

programme_part1_3.c

```c
// Premier exemple de programme simple

/*
 Ce commentaire ne modifie ni la taille,
 ni les performances du fichier exécutable (a.out)
*/

#include <stdio.h>

int main()
{
    printf("vive la physique\n");
    return 0;
}
```

programme_part1_4.c

```c
#include <stdio.h>


int main()
{
    printf("\n\n");
    printf("*************************************************\n");
    printf("**             Laurent de FORGES               **\n");
    printf("**                                             **\n");
    printf("**       Institut Non Lineaire de Nice         **\n");
    printf("**    Universite de Nice Sophia-Antipolis      **\n");
    printf("*************************************************\n");
    printf("\n\n");

    return 0;
}
```

Partie 2

---

programme_part2_1.c

```c
#include <stdio.h>

int main()
{
    // declaration des variables
    int i;
    float x;
    double y;
    char c;

    // affectations des variables
    i=2;
    x=4.2;
    y=3.14159;
    c='a';

    // impression des variables
    printf("i vaut : %d\n",i);
    printf("x vaut : %f\n",x);
    printf("y vaut : %lf\n",y);
    printf("c vaut : %c\n",c);

    return 0;
}
```

programme_part2_2.c

```c
// Attention : ce programme est un exemple de ce qu il ne faut pas faire !

#include <stdio.h>

int main()
{
    // declaration des variables
    int i, j;

    // affectations des variables
    i=2, j=5;

    // impression des variables
    printf("i vaut : %c\n",i);
    printf("j/i vaut : %i\n",j/i);
    printf("j/i vaut : %f\n",j/i);

    return 0;
}
```

programme_part2_2_corrige.c

```c
// Attention : ce programme est une correction de programme_part2_2.c

#include <stdio.h>

int main()
{
    // declaration des variables
    int i, j;

    // affectations des variables
    i=2, j=5;

    // impression des variables
    printf("i vaut : %d\n",i);
    printf("j/i vaut : %i\n",j/i);
    printf("j/i vaut : %f\n",(float) j/i);

    return 0;

}
```

programme_part2_3.c

```c
#include <stdio.h>

// declaration des constantes avec un type precise
// ces constantes peuvent etre utilisees dans main et en dehors
const int i=3;
const float x=4.111;

// declaration des constantes avec un type implicitement precise
// ces constantes peuvent etre utilisees dans main et en dehors
#define PI 3.141592
#define N 10

int main()
{

    // impression des constantes
    printf("i vaut : %d\n",i);
    printf("x vaut : %f\n",x);
    printf("PI vaut : %f\n",PI);
    printf("N vaut : %d\n",N);

    return 0;
}
```

Partie 3

---

programme_part3_1.c

```c
#include <stdio.h>

int main()
{
    // declaration des variables
    int i, j;

    // affectations des variables via la console
    printf("Entrez deux nombres entiers i et j:\n");
    scanf("%d%d",&i,&j);

    // impression des variables
    printf("i+j=%d\n",i+j);
    printf("i-j=%d\n",i-j);
    printf("i*j=%d\n",i*j);
    printf("i/j=%f\n",(float)i/j);

    return 0;
}
```

programme_part3_2.c

```c
#include <stdio.h>

int main()
{
    // declaration des variables
    int i, j;

    // affectations des variables via la console
    printf("Entre ton jour et moi de naissance :\n");
    scanf("%d%d",&i,&j);

    // impression des variables
    printf("Il te reste %d jours a vivre !\n",i*j);
    printf("PROFITE UN MAX DE TA VIE !!!\n");

    return 0;
}
```

programme_part3_3.c

```c
#include <stdio.h>

int main()
{

    FILE * file1; // pointeur de type FILE pour lire ici
    FILE * file2; // pointeur de type FILE pour ecrire ici

    // declaration des variables (decimales)
    double x1, y1, z1;
    double x2, y2, z2;

    // pour lire dans le fichier existant mesdonnees_input.txt
    file1=fopen("mesdonnees_input.txt","r");

    // pour creer et ecrire dans le fichier mesdonnees_ouput.txt
    file2=fopen("mesdonnees_ouput.txt","w");

    // affectation des valeurs des nombres de la ligne 1 aux variables x1,y1,z1
    fscanf(file1,"%lf%lf%lf",&x1,&y1,&z1);

    // affectation des valeurs des nombres de la ligne 2 aux variables x2,y2,z2
    fscanf(file1,"%lf%lf%lf",&x2,&y2,&z2);

    // impression dans un fichier texte
    fprintf(file2,"%lf  %lf\n",x1,y1+z1);
    fprintf(file2,"%lf  %lf\n",x2,y2+z2);

    // on ferme les fichiers dans lesquels on a lu et ecrit
    fclose(file1);
    fclose(file2);

   return 0 ;
}
```

mesdonnees_input.txt

```
1.0   10.0   100.0
2.0   20.0   200.0
```

Partie 4

---

programme_part4_1.c

```c
#include <stdio.h>
#include <math.h>      // on inclut les librairies pour utiliser sqrt et exp

#define PI 3.141592    // declaration de la constante PI


int main()
{

    // declaration des variables
    float xmin, xmax, x;  // bornes de l intervalle et variable de la gaussienne
    float norm;           // norme de la gaussienne


    // i: variable muette de la boucle for; Nombrepoints: nombre total de points
    int i, Nombrepoints;


    // initialisation des variables
    xmin=-5;
    xmax=5;
    norm=1/sqrt(2*PI);
    Nombrepoints=200;


    // la boucle commence pour i=0 et finit en i=Nombrepoints
    // on incremente i de 1 a chaque pas
    for(i=0;i<=Nombrepoints;i++)
    {

          // x augmente de (xmax-xmin)/Nombrepoints à chaque pas
          x=xmin+i*(xmax-xmin)/Nombrepoints;
          // on imprime x et la valeur de la gaussienne correspondante
          printf("%f    %f\n",x,norm*exp(-x*x/2));


    }   // fin de la boucle quand i=Nombrepoints


    return 0;
}
```

programme_part4_2.c

```c
#include <stdio.h>
#include <math.h>      // on inclut les librairies pour utiliser sqrt et exp

#define PI 3.141592    // declaration de la constante PI

int main()
{

    // declaration d un pointeur de type FILE pour ecrire dans un fichier
    FILE *file;
    // pour creer le fichier gauss_output.txt
    file=fopen("gauss_output.txt","w");


    // declaration des variables
    // bornes de l intervalle et variable de la gaussienne
    float xmin, xmax, x;
    // norme de la gaussienne
    float norm;
    // i: variable muette de la boucle for; Nombrepoints: nombre total de points
    int i, Nombrepoints;


    // initialisation des variables
    xmin=-5;
    xmax=5;
    norm=1/sqrt(2*PI);
    Nombrepoints=200;


    // la boucle commence pour i=0 et finit en i=Nombrepoints
    // on incremente i de 1 a chaque pas
    for(i=0;i<=Nombrepoints;i++)
    {

          // x augmente de (xmax-xmin)/Nombrepoints a chaque pas
          x=xmin+i*(xmax-xmin)/Nombrepoints;
          // on imprime x et f(x) dans le fichier gauss_output.txt
          fprintf(file,"%f    %f\n",x,norm*exp(-x*x/2));


    } // fin de la boucle quand i=Nombrepoints


    // il faut refermer le fichier cree
    fclose(file);

    return 0;
}
```

programme_part4_3.c

```c
#include <stdio.h>

// on inclut les librairies pour utiliser sqrt et exp
#include <math.h>

// declaration de la constante PI
#define PI 3.141592

int main()
{
    // declaration d un pointeur de type FILE pour ecrire dans un fichier
    FILE *file;

    // pour creer le fichier gauss_output.txt
    file=fopen("gauss_output.txt","w");

    // declaration des variables
    // bornes de l intervalle et variable de la gaussienne
    float xmin, xmax, x;
    // norme et ecart type de la gaussienne
    float norm, sigma;
    // i: variable muette de la boucle for; Nombrepoints: nombre total de points
    int i, Nombrepoints;

    printf("Entrez la valeur de sigma :\n");
    // lecture de l ecart type dans la console
    scanf("%f",&sigma);

    // initialisation des variables
    xmin=-5;
    xmax=5;
    norm=1/sqrt(2*PI*sigma*sigma);
    Nombrepoints=200;

    // la boucle commence pour i=0 et finit en i=Nombrepoints
    // on incremente i de 1 a chaque pas
    for(i=0;i<=Nombrepoints;i++)
    {
            // x augmente de (xmax-xmin)/Nombrepoints a chaque pas
            x=xmin+i*(xmax-xmin)/Nombrepoints;
            // on imprime x et f(x) dans le fichier gauss_ouput.txt
            fprintf(file,"%f  %f\n",x,norm*exp(-x*x/(2*sigma*sigma)));

    } // fin de la boucle quand i=Nombrepoints

    // il faut refermer le fichier cree
    fclose(file);

    return 0;
}
```

programme_part4_4.c

```c
#include <stdio.h>

// on inclut les librairies pour utiliser sqrt et exp
#include <math.h>

// declaration de la constante PI
#define PI 3.141592

int main()
{

    // declaration des variables
    // bornes de l intervalle et variable de la gaussienne
    float xmin, xmax, x;
    // norme de la gaussienne
    float norm;
    // i: variable muette de la boucle for; Nombrepoints: nombre total de points
    int i, Nombrepoints;


    // initialisation des variables
    xmin=-5;
    xmax=5;
    norm=1/sqrt(2*PI);
    Nombrepoints=200;
    i=0;

    // la boucle commence pour i=0 et finit en i=Nombrepoints
    // on incremente i de 1 a chaque pas
    while(i<=Nombrepoints)
    {

            // x augmente de (xmax-xmin)/Nombrepoints a chaque pas
            x=xmin+i*(xmax-xmin)/Nombrepoints;

            // on imprime x et la valeur de la gaussienne correspondante
            printf("%f  %f\n",x,norm*exp(-x*x/2));

            // on incremente i d une unite
            i=i+1;

    }   // fin de la boucle quand i=Nombrepoints


    return 0;
}
```

programme_part4_5.c

```c
#include <stdio.h>

// on inclut les librairies pour utiliser sqrt et exp
#include <math.h>

// declaration de la constante PI
#define PI 3.141592

int main()
{

    // declaration des variables
    // bornes de l intervalle et variable de la gaussienne
    float xmin, xmax, x;
    // norme de la gaussienne
    float norm;
    // i: variable muette de la boucle for; Nombrepoints: nombre total de points
    int i, Nombrepoints;


    // initialisation des variables
    xmin=-5;
    xmax=5;
    norm=1/sqrt(2*PI);
    Nombrepoints=200;
    i=0;

    // la boucle commence avec i=0
    // elle incremente i de 1 a chaque pas et tourne tant que i<=Nombrepoints
    // on executera toujours au moins 1 fois l instruction
    // meme si i>=Nombrepoints
    do
    {

         // x augmente de (xmax-xmin)/Nombrepoints a chaque pas
         x=xmin+i*(xmax-xmin)/Nombrepoints;

         // on imprime x et la valeur de la gaussienne correspondante
         printf("%f     %f\n",x,norm*exp(-x*x/2));

         // on incremente i d une unite
         i=i+1;

    }   // fin de la boucle quand i=Nombrepoints

    while(i<=Nombrepoints);

    return 0;
}
```

programme_part4_6.c

```c
#include <stdio.h>

int main()
{
	int i, j;
	int imax=7, jmax=5;

	for(i=1;i<=imax;i++)
	{
		for(j=1;j<=jmax;j++)
		{
			printf("c%d%d	",i,j);
		}
		printf("\n");
	}
	return 0;
}
```

programme_part4_7.c

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

int main()
{

	int i,N;
	srand48(time(NULL));

	i=10*drand48();

	printf("Entrez un nombre entre 0 et 10 :\n");
	scanf("%d",&N);

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

	if(N==i)
	{
		printf("WAW ... VOUS AVEZ GAGNE\n");
	}


    return 0;
}
```

programme_part4_8.c

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


int main()
{

	int i,N;
	srand48(time(NULL));

	i=10*drand48();

	printf("Entrez un nombre entre 0 et 10 :\n");
	scanf("%d",&N);


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


	// si N et i sont egaux, le programme execute l insctruction
	if(N==i)
	{
		printf("WAW ... VOUS AVEZ GAGNE\n");
	}



	// si N et i ne sont pas egaux, le programme execute l insctruction
	else
	{
		printf("PERDU ... RECOMMENCEZ\n");
	}


    return 0;
}
```

programme_part4_9.c

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


int main()
{

	// declaration des variables
	double x, y, R, Pi;
	int i, n, N;

	srand48(time(NULL));

	// initialisation des variables
	N=100000000;
	n=0;

	// on utilise une boucle pour repartir les points dans le carre
	for(i=0;i<N;i++)
	{

		x=drand48();
		y=drand48();
		// on calcul la distance entre le point tire et l origine
		R=sqrt(x*x+y*y);

		// on test si ce point se situe dans le quart de cercle
		if(R<=1)
		{
			// on comptabilise les points dans le quart de cercle
			n++;
		}
	}

	// on calcul pi en faisant attention d avoir un nombre decimale
	Pi=4*(float)n/(float)N;

	// on imprime a l ecran la valeur de pi
	printf("Pi=%f\n",Pi);


    return 0;
}
```

Partie 5

---

programme_part5_1.c

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

int main()
{

    FILE *file;
    // ouverture d un fichier de donnees dans lequel on va ecrire
    file=fopen("out.txt","w");

    // declaration des variables
    // temps caracteristique de relaxation et pas d integration
    double tau=1.0, delta=0.01;
    // bornes de l intervalle du temps d integration
    double a=0,b=5*tau;
    // variable muette pour la boucle et nombre d etapes
    int i, N=(int)(b-a)/delta;
    // temps et vitesse (tableau a N+1 cases)
    double t, v[N];


    // initialisation des variables
    v[0]=1.0;   // vitesse initiale
    t=0.0;      // temps initial

    for(i=0;i<N;i++)   // boucle d integration de la vitesse du bateau
    {
            // le temps s ecoule de delta
            t=t+delta;
            // calcul de la vitesse au pas suivant (i+1)
            v[i+1]=v[i]*(1-delta/tau);

            // impression du temps et de la vitesse
            fprintf(file,"%lf %lf\n",t,v[i+1]);
    }

    fclose(file);

    return 0;
}
```

programme_part5_2.c

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

int main()
{

    FILE *file;
    // ouverture d un fichier de donnees dans lequel on va ecrire
    file=fopen("out.txt","w");


    // declaration des variables
    // taux de natalite, de mortalite et pas d integration
    double alpha=1.0, beta=0.5, delta=0.01;
    // bornes de l intervalle du temps d integration
    double a=0,b=10;
    // variable muette pour la boucle et nombre d etapes
    int i, N=(int)(b-a)/delta;
    // temps et nombre de coccinelles(*100) (tableau a N+1 cases)
    double t, n[N];


    // initialisation des variables
    n[0]=5.0;   // nombre initial de coccinelles(*100)
    t=0.0;      // temps initial

    for(i=0;i<N;i++)   // boucle d integration de la vitesse du bateau
    {

            // le temps s ecoule de delta
            t=t+delta;

            // calcul du nombre de coccinelles au pas suivant (i+1)
            n[i+1]=n[i]*(1+alpha*delta-beta*delta*n[i]);

            // impression du temps et du nombre de coccinelles
            fprintf(file,"%lf %lf\n",t,n[i+1]);

    }

    fclose(file);

    return 0;
}
```

programme_part5_3.c

```c
#include <stdio.h>

// temps caracteristique de relaxation
# define tau 1.0

// pas d integration
# define delta 0.01

// fonction qui a comme argument un nombre decimale v
// elle renvoie un nombre decimale f(v)
double f(double v)
{
         // la fonction calcul la valeur v*(1-delta/tau) et la renvoie
         return v*(1-delta/tau);
}

int main()
{

    // declaration des variables
    // bornes de l intervalle du temps d integration
    double a=0,b=5*tau;
    // variable muette pour la boucle et nombre d etapes
    int i, N=(int)(b-a)/delta;
    // temps et vitesse (tableau a N+1 cases)
    double t, v[N];

    // initialisation des variables
    v[0]=1.0;   // vitesse initiale
    t=0.0;      // temps initial


    // boucle d integration de la vitesse du bateau
    for(i=0;i<N;i++)
    {

            // le temps s ecoule de delta a chaque pas
            t=t+delta;

            // calcul de la vitesse au pas suivant (i+1) grace a la fonction f
            v[i+1]=f(v[i]);

            // impression du temps et de la vitesse
            printf("%lf %lf\n",t,v[i+1]);

    }

    return 0;
}
```