Sup Galilée
Mathématiques Appliquées et Calcul Scientifique
Deuxième année
Année universitaire 2018–2019

Langage C++(G4SCCP)
Cours : X. Juvigny
TPs : S. Landier
sam.landier@onera.fr

Travaux pratiques N° 1

Les bases du langage C++

https://github.com/Macs1718/Promotion_2020
https://en.cppreference.com/w/
https://stackoverflow.com/

Exercice 1 (Pointeurs et Références)

a) Créer un fichier avec le suffixe exo1.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)

Dans la fonction main :

b) Définir une variable *i* de type int et y assigner la valeur 3
c) Ecrire l'instruction permettant d'afficher la valeur de i, et imprimer le résultat (i.e. compiler et exécuter le programme).
d) Modifier le flux de sortie pour obtenir l'affichage : « la valeur de i est : 3 »

e) Définir une variable *pi* de type pointeur de int.
f) Assigner à pi l'adresse de i et imprimer l'adresse de i en utilisant pi.
g) Modifier la valeur de i en utilisant pi et imprimer sa nouvelle valeur.

h) Définir une variable *ri* de type référence de int et compiler. Qu'obtient-on ? pourquoi ?
i) Corriger le problème en utilisant i.
j) Modifier la valeur de i en utilisant ri et imprimer le résultat.

Exercice 2 (Fonctions et le passage d'arguments)

Créer un fichier avec le suffixe exo2.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)

a) (Passage d'arguments par valeur)

Ecrire une fonction *swap1* qui prend 2 entiers a et b comme arguments, qui ne retourne rien, et qui

- imprime les valeurs et adresses des arguments
- permute les valeurs des arguments
- imprime de nouveau les valeurs

Dans la fonction main :

- définir et valuer 2 entiers a et b
- imprimer leurs valeurs et leurs adresses
- appeler la fonction swap1avec ces entiers
- réimprimer les valeurs et adresses de a et b.
- Qu'observe-ton ? Pourquoi ?

b) (Passage d'arguments par pointeur)

Reprendre les étapes de a) en définissant de façon analogue une fonction swap2 où les arguments sont passés cette fois par adresse. Pour les impressions internes, on veillera à afficher les infos (valeurs et adresses) des variables pointées plutôt que des pointeurs eux-mêmes.

c) (Passage d'arguments par référence)

Idem avec une troisième fonction swap3 où les arguments sont passés par référence.

Exercice 3 (boucles)

Ces exercices se basent sur l'emploi de la fonction d'entrée *std::cin* (toujours de <iostream>) qui permet lors de l'exécution de demander des valeurs à fournir par l'utilisateur.

Créer un fichier avec le suffixe exo3.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)

a) (boucle *for*)
Ecrire une fonction (et l'exécuter dans un main) *loop1* qui demande à l'exécution d'entrer une valeur à l'utilisateur et qui affiche, après chaque entrée, le cumul courant des valeurs. Le programme doit terminer après avoir entré 5 valeurs.

Le programme doit fournir une sortie du type :
« Entrez une valeur : »
5
« Le cumul est 5 »
« Entrez une valeur : »
7
« Le cumul est 12 »
…

b) (boucle *while*)
Dans l'exercice a) le nombre d'entrée est défini par le programme (5 entiers) et non l'utilisateur. Pour permettre à l'utilisateur de le choisir, reprendre l'exercice a) avec une boucle while à définir dans une nouvelle fonction *loop2*.

Pour éviter le problème d'une boucle infinie, définir une valeur d'arrêt permettant de sortir de la boucle.

c) (boucle *do while*)
Modifier l'exercice b) de manière à employer une boucle do while, nommer la fonction *loop3*.

Exercice 4 (Tableaux statiques et dynamiques fondamentaux)

Créer un fichier avec le suffixe exo4_5.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)

Tableaux statiques C

a) Définir un tableau statique t de 10 entiers (langage C)
b) Ecrire une fonction *init* qui prend le tableau en paramètre (pas par valeur !) et sa taille N et effectue une boucle qui value le tableau de N-1 à 0 : t[0] = N-1, t[1] =N-2…
et imprime les valeurs.
c) Imprimer l'adresse du 5ème élément avec l'opérateur.
d) Imprimer la valeur du 5ème élément à partir de l'adresse du premier élément et l'opérateur de déréférencement * . arithmétique de pointeurs).

Tableaux dynamiques C++

e) Allouer un tableau dynamique de taille 10 (opérateur new)
f) Appeler init avec ce tableau
g) Désallouer le tableau avec l'opérateur delete [] (important ! sinon cela engendre ce qu'on appelle une fuite mémoire)

Exercice 5 (Tableaux statiques et dynamiques de la STL)

Tableau statique C++11 : std ::array

a) Refaire a),b) et c) de l'exercice 3) avec un std::array de 10 entiers.
Nommer la fonction init_array.

Container STL : std::vector

b) Définir une fonction init_vector et l'exécuter:
- qui prend en paramètre un std::vector v et un entier N (taille souhaitée)
- qui redimensionne v à la taille N
- qui affecte les valeurs v[0]=N-1, …v[N-1]=0
- qui imprime les valeurs

c) Utiliser alors init avec v.

d) Définir et utiliser une fonction init_vector2 qui effectue les mêmes tâches avec l'aides d'un itérateur de vector (std ::vector<int> ::iterator)
e) Définir et utiliser une fonction init_vector3, avec la syntaxe de boucle C++11 et le mot clé *auto* en lieu et place de l'itérateur.