|  |
| --- |
|  |
| LA CRÉATION DE SES PROPRES TYPES  LES STRUCTURES |
| Module 2  420-C21-IN Programmation II Godefroy Borduas – Automne 2021 |

1

|  |
| --- |
|  |
| PLAN DE LEÇON |
| * Qu'est-ce qu'une structure ? * Comment définir une structure ? * Existe-t-il une convention de codage ? * Comment utiliser une structure ? * Utiliser cin ou cout sur les structures * L'impact sur le dictionnaire de donnée * Les structures dans des structures (Inception) * Utiliser des tableaux de structure * Exercices |

2

|  |
| --- |
|  |
| QU’EST-CE QU’UNE STRUCTURE ? |
|  |

3

|  |
| --- |
|  |
| ÇA TOUCHE LA MÉMOIRE |
| * Un des problèmes importants est l'organisation de l'information * En C11, vous avez vu les types « simples » et les types « agrégats »   + Type simple : Type de variable de base (nombre, caractères)   + Type agrégats : Type réunissant plusieurs valeurs de type simple (tableau, string) * Très utile pour des problèmes simples, exemple :   + Calcul sur des entiers (trois entiers)   + Collecter les notes d'une classe (tableau de notes) |

4

|  |
| --- |
|  |
| MAINTENANT, PRENONS UNE INFORMATION PLUS COMPLEXE |
| * Le profil d'une personne étudiante :   + Nom   + Prénom   + Matricule   + Téléphone   + Nombre de cours contributoire   + Moyenne générale * Maintenant, créez un programme informatique afin de gérer le profil pour une personne étudiante   + Modifier le téléphone   + Modifier la moyenne générale |

5

|  |
| --- |
|  |
| QU’AVEZ-VOUS REMARQUÉ ? |
|  |

6

|  |
| --- |
|  |
| ALLONS-Y AVEC UNE EXPÉRIENCE DE PENSÉE |
| * Imaginez que le système doit gérer, en tout temps, dix personnes étudiantes * Comment allez-vous modifier le code ? * Qu’est-ce que vous en concluez ? * Intuitivement, y a-t-il une solution ? |

7

|  |
| --- |
|  |
| PENSONS EN TERMES D’ENTITÉ |
| * Et si la personne étudiante était une entité informatique   + Entité : Élément informatique manipulable et traitable (ex. int, float, etc.) * Dans ce cas, la personne étudiante aura des **sous-entités** * En langage informatique, on parle de **structure** |

8

|  |
| --- |
|  |
| DÉFINITION |
| Une **structure** est un type informatique composé ou dérivé des types simples. Ce type permet de manipuler un ensemble d’information comme un seul objet.  Similaire à un tableau, la structure réunit au même endroit toutes les informations relatives à une même entité. |

9



|  |
| --- |
|  |
| À RESSEMBLE UNE STRUCTURE |
| * Revenons à la personne étudiante, il est possible de créer une structure pour réunir toutes les informations au même endroit. * La structure ressemble à ceci   Personne étudiante Nom (string) Prénom (string) Matricule (int) Téléphone (long)  Nombre de cours contributoire (unsigned int) Moyenne générale (float) |

10

|  |
| --- |
|  |
| COMMENT DÉFINIR UNE STRUCTURE ? |
|  |

11

|  |
| --- |
|  |
| GRAMMAIRE D’UNE DÉFINITION D’UNE STRUCTURE EN C++ |
| * C++ permet la définition directe des structures * La forme générale   struct NomDeLaStructure\_s  {  type NomDuMembre\_1; type NomDuMembre\_2;  …  type NomDuMembre\_N;  };   * Les types simples (ou d’agrégat) qui forment la structure sont appelés des   **membres** de la structure.   * Tous les noms de structure doivent se terminer par **\_s** (convention) |

12

|  |
| --- |
|  |
| À QUOI RESSEMBLE LA DÉFINITION DE LA STRUCTURE PERSONNE ÉTUDIANTE |
| **Planification de notre structure** **Version C++ de notre structure**  Personne étudiante struct PersonneEtudiante\_s  Nom (string) {  Prénom (string)  string Nom;  Matricule (int)  Téléphone (long) string Prenom;  Nombre de cours contributoire int Matricule;  (unsigned int)  Moyenne générale (float) long Telephone;  usigned int CoursContri; float MoyenneGenerale;  }; |

13

|  |
| --- |
|  |
| OÙ DOIT-ON PLACER LA DÉFINITION |
| * La définition va toujours avant la fonction *main*   + Juste après les instructions de préprocesseurs et using namespace std * S’il y en a trop, on place ces définitions dans un fichier d’en-tête (\*.h)   + Nous verrons dans quelques cours ce que ça signifie |

14

|  |
| --- |
|  |
| À QUOI RESSEMBLE NOTRE EXEMPLE DE LA PERSONNE ÉTUDIANTE |
| #include <iostream> int main()  using namespace std; { struct PersonneEtudiante\_s …  { }  string Nom; string Prenom; int Matricule; long Telephone;  unsigned int CoursContri; float MoyenneGenerale;  }; |

15

|  |
| --- |
|  |
| EXISTE-T-IL UNE CONVENTION DE CODAGE ? |
| Simplement, oui ! |

16

|  |
| --- |
|  |
| 5 RÈGLES DE CODAGE |
| * Ce ne sont pas des règles du langage. Ce sont des ententes entre les programmeuses et les programmeurs pour faciliter la lecture et la maintenance des logiciels * Les cinq règles doivent être observées en tout temps   1. Les structures doivent être déclarées **avant** la fonction *main*   2. Lorsqu’il y a plus de cinq structures, ces dernières doivent être écrites dans un fichier d’en-tête (\*.h)   3. Le nom (identifiant) d’une structure doit toujours se terminer par **\_s**   4. Les variables du type de la structure doivent être déclarées dans une fonction soit la fonction *main* ou une autre fonction   5. Les déclarations de variable de type de la structure au niveau globale devront être justifiées      + Nous expliquerons ce que ça signifie dans quelques cours |

17

|  |
| --- |
|  |
| COMMENT UTILISER UNE STRUCTURE ? |
|  |

18

|  |
| --- |
|  |
| PREMIÈREMENT, DÉCLARONS UNE VARIABLE |
| * La définition d’une structure permet de la décrire l’entité * Pour l’utiliser, il faut **déclarer** une variable du type de la structure   + La définition se fait comme toutes les autres types   + Forme générale :   <Type\_structure> <identifiant>;   * + Dans notre exemple :   PersonneEtudiante\_s LaPersonne;   * C’est à ce moment que le compilateur procède à l’allocation de l’espace mémoire   + Maintenant la structure existe dans la RAM * À ce moment, les valeurs de chaque membre correspondront aux valeurs par défaut |

19

|  |
| --- |
|  |
| COMMENT SPÉCIFIER UNE VALEUR LORS DE L’INITIALISATION ? |
| * On peut spécifier les valeurs initiales de la variable avec la même syntaxe que les tableaux   + La position de la valeur correspond à la position de la déclaration du membre   + Forme générale   <Type\_structure> <identifiant> = { <valeur\_membre1>, <val\_mem2>, …, <val\_memN>};   * + Dans notre exemple :   PersonneEtudiante\_s personne = {"Borduas", "Godefroy", 21223366, 5797214447, 2 + 3,  42.0};   * + L’ordre des valeurs est donc :   {Nom, Prenom, Matricule, Telephone, CoursContri, MoyenneGenerale} |

20

|  |
| --- |
|  |
| ACCÉDER AUX MEMBRES D’UNE STRUCTURE |
| * L’accès aux membres permet de :   + Lire le contenu (sa valeur)   + Écrire du contenu * L’accès (lecture/écriture) se fait via l’**opérateur point** (**.**) placer entre la variable de type de la structure et le membre   + Forme générale :   <variable>.<identifiant du membre>   * + Dans notre exemple :   personne.nom   * + Signifie :   Nom **élément de** personne |

21

|  |
| --- |
|  |
| AFFECTER UNE VALEUR AU MEMBRE |
| * L’affectation se fait grâce à l’opérateur d’affectation (**=**)   + Forme générale :   <variable>.<identifiant du membre> = <valeur>;   * + Dans notre exemple :   personne.Telephone = 5149823437; |

22

|  |
| --- |
|  |
| LIRE LE CONTENU D’UN MEMBRE |
| * Pour utilise le membre (via l’opérateur point) comme tout autre variable * Prenons l’exemple du nouveau cours   + Premier code sans structure   Sn = MoyenneGen \* CoursContributoire;   * + Deuxième code avec structure   Sn = personne.MoyenneGenerale \* personne.CoursContri; |

23

|  |
| --- |
|  |
| REPRENONS L’EXERCICE DU DÉBUT DU COURS |
|  |

24

|  |
| --- |
|  |
| MAINTENANT, PENSEZ AU PROGRAMME AVEC 10 PERSONNES ÉTUDIANTES |
|  |

25

|  |
| --- |
|  |
| UTILISER CIN OU COUT SUR LES STRUCTURES |
|  |

26

|  |
| --- |
|  |
| CIN ET COUT LISSENT LES TYPES SIMPLES SEULEMENT |
| * Impossible de lire directement un type de structure * Doit être appelé sur un membre seulement   cin >> personne.Telephone;  cout << personne.MoyenneGenerale << endl; |

27

|  |
| --- |
|  |
| L'IMPACT SUR LE DICTIONNAIRE DE DONNÉE |
|  |

28

|  |
| --- |
|  |
| ON DÉTAILLE CHAQUE MEMBRE |
| * Comme les types simples, on doit donner l’identifiant de la variable, la signification du type et son type * Il en va de même pour les membres * La valeur est donnée par membre |

29

|  |  |  |  |
| --- | --- | --- | --- |
| **Idenficateur** | **Signification** | **Type** | **Valeur** |
| Personne | Structure représentant les informations d’une personne étudiante | PersonneEtudiante\_s |  |
| Nom | Nom de la personne étudiante | string | "Borduas" |
| Prenom | Prénom de la personne étudiante | string | "Godefroy" |
| Matricule | Identifiant numérique de la personne étudiante | int | 21223366 |
| Telephone | Numéro de téléphone de la personne étudiante | long | 5797214447 |
| CoursContri | Nombre de cours contribuant à la moyenne générale | unsigned int | 5 |
| MoyenneGeneral | Moyenne générale de la personne étudiante | float | 42.0 |

|  |
| --- |
|  |
| LES STRUCTURES DANS DES STRUCTURES (INCEPTION) |
|  |

30

|  |
| --- |
|  |
| UNE STRUCTURE PEUT CONTENIR TOUS LES TYPES |
| * Il est possible d’ajouter un type de structure dans un autre type de structure * Exemple, le numéro de téléphone est une structure   Telephone\_s  IdentifiantRegional IdentifiantLocal IdentifiantUnique   * + Dans le cas du numéro 579 721-4447   Telephone\_s  IdentifiantRegional = 579  IdentifiantLocal = 721  IdentifiantUnique = 4447 |

31

|  |
| --- |
|  |
| UNE FOIS CRÉÉ, ON L’AJOUTE DANS LA STRUCTURE |
| * Après avoir défini la structure imbriquée, on l’ajoute dans notre structure principale :   struct Telephone  {  int IdentifiantRegionale; struct PersonneEtudiante\_s int IdentifiantLocale; {  int IdentifiantUnique; string Nom;  }; string Prenom;  int Matricule;  **Telephone\_s Telephone**; unsigned int CoursContri; float MoyenneGenerale;  }; |

32

|  |
| --- |
|  |
| COMMENT ACCÉDER AUX MEMBRES DE LA STRUCTURE IMBRIQUÉE ? |
| * Il faut passer par le membre contenant la structure imbriquée et ensuite accéder au membre souhaiter * Toujours utiliser l’opérateur point * Exemple pour accéder à IndentifiantRegionale :   personne.Telephone.IdentifiantRegionale   * + Mode lecture   cout << personne.Telephone.IdentifiantRegionale;   * + Mode écriture   personne.Telephone.IdentifiantRegionale = 514; |

33

|  |
| --- |
|  |
| UTILISER DES TABLEAUX DE STRUCTURE |
|  |

34

|  |
| --- |
|  |
| C’EST COMME LES TYPES SIMPLES |
| * Pour créer un tableau d’un type de structure, on utilise la même syntaxe que pour les types simples   + Forme générale :   <type de structure> <identifiant>[<taille>];   * + Dans notre exemple :   PersonneEtudiante\_s personnes[10];   * Par la suite, le compilateur traite les tableaux de structure comme tout autre type de tableau * Leur fonctionnement est identique |

35

|  |
| --- |
|  |
| LECTURE ET AFFECTATION DES VALEURS |
| * Lecture :   cout << NomTableau[indice];   * Affectation :   cin >> NomTableau[indice]; |

36

|  |
| --- |
|  |
| EXERCICES |
|  |

37

* Créer une structure qui représente un point dans un plan
* Demander à la personne utilisatrice d’entrée deux points et calculer sa distance entre ces points
  + Calcul de la distance entre deux points

𝑑i𝑠𝑡𝑎𝑛𝑐e = 𝑥2 − 𝑥1 2 + 𝑦2 − 𝑦1 2

LES POINTS DANS UN PLAN

|  |  |  |  |
| --- | --- | --- | --- |
| **Idenficateur** | **Signification** | **Type** | **Valeur** |
| Point | Structure représentant un point dans l’espace | *Choisissez* |  |
| x | Position réelle (nombre flottant) sur l’axe des abscisses (gauche- droite) | *Choisissez* | "Borduas" |
| Y | Position réelle (nombre flottant) sur l’axe des ordonnées (haut- bas) | *Choisissez* | "Godefroy" |

38

|  |
| --- |
|  |
| FONCTIONS MATHÉMATIQUES UTILES |
| * Calcul d’une racine carrée   + Dans la bibliothèque : <cmath>   + Prototype :     - double Resultat = sqrt(Valeur);     - float Resultat = sqrt(Valeur);   + https://en.cppreference.com/w/cpp/numeric/math/sqrt * Calcul d’une puissance carrée   + Dans la bibliothèque : <cmath>   + Prototype :     - double Resultat = pow(Valeur, 2);     - float Resultat = pow(Valeur, 2);   + https://en.cppreference.com/w/cpp/numeric/math/pow |

39

|  |
| --- |
|  |
| MANIPULER DES TABLEAUX D’ÉTUDIANT |
| * Écrire un programme C, qui lit les noms complets des étudiants et leurs moyennes dans un tableau de structures. Puis actualise ces moyennes en ajoutant un bonus de:   + *1* point pour les étudiants ayant une note strictement inférieure à *10*.   + *0.5* point pour les étudiants ayant une note entre *10* et *15* incluse.   *N.B.*: la structure doit avoir deux éléments: une chaîne de caractères et un réel. |

40

|  |
| --- |
|  |
| LES VILLES ET LES HABITANTS |
| * Écrire un programme C, qui lit un ensemble de villes avec leur nombre d'habitants dans un tableau de structures, les trie et les affiche dans l'ordre croissant des nombres d'habitants. |

41

|  |  |  |  |
| --- | --- | --- | --- |
| **Idenficateur** | **Signification** | **Type** | **Valeur** |
| Ville | Structure représentant une ville | *Choisissez* |  |
| Nom | Nom de la ville | *Choisissez* | "Borduas" |
| NbreHabitant | Nombre d’habitant | *Choisissez* | "Godefroy" |

|  |
| --- |
|  |
| UTILISATION DE *TYPEDEF* |
| Petite info gratuite I |

42

|  |
| --- |
|  |
| DÉFINITION D’UN TYPE |
| * Nous avons vu la définition d’une structure   struct IdentifiantDeLaStructure { … };   * Dans le langage C++, cette déclaration définit un nouveau type qui se nomme   **IdentifiantDeLaStructure**   * Toutefois, ce n’est pas valable en C pur. * La définition précédente définit seulement une structure en C pur et non un type.   + Le type est alors **struct IdentifiantDeLaStructure** * Pour définir le type, l’instruction est alors :   Typedef struct IdentifiantDeLaStructure IdentifiantDeLaStructure\_s |

43

|  |
| --- |
|  |
| DÉFINIR DIRECTEMENT LA STRUCTURE |
| * On peut définir directement la structure   typedef struct {  …  } IdentifiantDeLaStructure\_s; |

44

|  |
| --- |
|  |
| LES ÉNUMÉRATIONS |
| Petite info gratuite II |

45

|  |
| --- |
|  |
| QU’EST-CE QU’UNE ÉNUMÉRATION ? |
| * Au sens littéraire, l’énumération est une « énonciation successive des éléments d’un tout ». * L’énumération est donc une liste d’élément. * Exemple :   + Couleur : rouge, bleu, jaune et vert   + Forme : carré, rond, triangle et pentagone   + Fruit : pomme, fraise, tomate et banane * En programmation, l’énumération est une liste fixe.   + Elle ne peut être réduite   + Elle ne peut être allongée |

46

|  |
| --- |
|  |
| À QUOI SERVENT LES ÉNUMÉRATIONS ? |
| * À limiter les choix dans un programme * À faciliter la gestion des choix * Imaginez qu’une fonction offre trois choix pour fonctionner :   + Additionner les paramètres   + Soustraire les paramètres   + Multiplier les paramètres * L’énumération permet de limiter le choix à ces trois options * La fonction recevra simplement la valeur liée à l’énumération |

47

|  |
| --- |
|  |
| REPRÉSENTATION D’UNE ÉNUMÉRATION EN C |
| * L’énumération est un type entier déguisé * Chaque élément de l’énumération est représenté par un nombre entier * Pour nous, on utilise une étiquette au lieu du nombre * Par la suite, on utilise l’énumération comme un type |

48

|  |
| --- |
|  |
| DÉFINIR ET UTILISER L’ÉNUMÉRATION |
| * La définition d’une énumération à la forme suivante :   + Forme générale :   enum IdentifiantEnumeration { valeur1, valeur2, …, valeurN};   * + Exemple :   enum Couleur {jaune, rouge, bleu, vert};   * Affecter une énumération :   Couleur c1 = jaune;   * Cette action est interdite :   jaune = 3;  Jaune est une valeur dès quelle est défini dans une énumération |

49

|  |
| --- |
|  |
| DÉFINIR LA VALEUR DES ÉNUMÉRATIONS |
| * Comme les énumérations sont des listes d’entier, on peut définir la valeur d’un item * Pour définir la valeur, on affecte une valeur entière à l’étiquette * Exemple :   enum Couleur {jaune = 5, bleu, vert = 4, bleu = 42}; |

50

|  |
| --- |
|  |
| UTILISATION D’UNE ÉNUMÉRATION DANS UN CALCUL ENTIER |
| * Comme l’énumération est un entier, on peut l’utiliser dans les calculs   int p;  enum Couleur {jaune = 5, bleu, vert = 4, bleu = 42}; Couleur c = jaune;  p = 2 + vert \* jaune;   * La valeur de p est alors 22, car vert = 4 et jaune = 5 |

51