Bing

Bonjour, je suis ravi que vous vous intéressiez à la programmation structurée. C'est un paradigme de programmation qui vise à rendre le code plus clair, plus modulaire et plus facile à maintenir. Je vais essayer de vous aider à concevoir un contenu de formation pour vos étudiants sur ce concept.

La programmation structurée repose sur trois principes fondamentaux :

* La \*\*séquence\*\*: il s'agit d'exécuter des instructions les unes après les autres, dans l'ordre où elles sont écrites.
* L'\*\*alternative\*\*: il s'agit de choisir entre deux ou plusieurs blocs d'instructions, en fonction d'une condition logique.
* La \*\*répétition\*\*: il s'agit de répéter un bloc d'instructions tant qu'une condition logique est vérifiée.

Ces trois principes permettent de construire des programmes sans utiliser l'instruction **goto**, qui est considérée comme nuisible à la lisibilité et à la fiabilité du code . Au lieu de cela, on utilise des structures de contrôle telles que **while**, **repeat**, **for**, **if .. then .. else**, etc.

La programmation structurée encourage également la décomposition du code en petits sous-programmes (appelés fonctions ou procédures), qui réalisent des tâches spécifiques et indépendantes. Ces sous-programmes doivent éviter d'utiliser des variables globales, qui peuvent provoquer des effets de bord indésirables. Ils doivent plutôt utiliser des variables locales et des paramètres, qui sont passés par valeur ou par référence .

Ces techniques permettent de créer du code plus facile à comprendre, à tester et à modifier. Elles facilitent également la conception par décompositions successives (top-down design), qui consiste à partir du problème général et à le diviser en sous-problèmes plus simples, jusqu'à atteindre le niveau des instructions élémentaires.

Voici un exemple de contenu que vous pourriez utiliser pour illustrer le concept de programmation structurée à vos étudiants:

**Exemple:**

On veut écrire un programme qui calcule la somme des n premiers entiers naturels, où n est saisi par l'utilisateur.

**Solution:**

On peut utiliser la programmation structurée pour résoudre ce problème. On commence par définir le problème général:

* Entrée: un entier naturel n
* Sortie: la somme des n premiers entiers naturels
* Traitement: calculer la somme en utilisant une boucle

On peut ensuite décomposer le problème en sous-problèmes plus simples:

* Entrée: lire n
* Sortie: afficher la somme
* Traitement: initialiser la somme à 0, initialiser un compteur à 1, répéter tant que le compteur est inférieur ou égal à n: ajouter le compteur à la somme, incrémenter le compteur

On peut enfin traduire ces sous-problèmes en instructions dans un langage de programmation. Par exemple, en pseudo-code:

// Lire n

Lire n

// Initialiser la somme à 0

somme <- 0

// Initialiser le compteur à 1

compteur <- 1

// Répéter tant que le compteur est inférieur ou égal à n

Tant que compteur <= n faire

// Ajouter le compteur à la somme

somme <- somme + compteur

// Incrémenter le compteur

compteur <- compteur + 1

Fin Tant que

// Afficher la somme

Afficher somme

Ce programme est structuré selon les trois principes de la programmation structurée: séquence, alternative et répétition. Il ne contient pas d'instruction goto. Il utilise des variables locales (somme et compteur) et un paramètre (n). Il pourrait être encapsulé dans une fonction ou une procédure.

J'espère que cet exemple vous a été utile. Si vous avez d'autres questions sur la programmation structurée, n'hésitez pas à me les poser. 😊

Hello, I am glad that you are interested in structured programming. It is a programming paradigm that aims to make the code clearer, more modular and easier to maintain. I will try to help you design a training content for your students on this concept.

Structured programming is based on three fundamental principles :

* The \*\*sequence\*\*: it consists of executing instructions one after the other, in the order in which they are written.
* The \*\*alternative\*\*: it consists of choosing between two or more blocks of instructions, depending on a logical condition.
* The \*\*repetition\*\*: it consists of repeating a block of instructions as long as a logical condition is verified.

These three principles allow to build programs without using the **goto** instruction, which is considered harmful to the readability and reliability of the code . Instead, control structures such as **while**, **repeat**, **for**, **if .. then .. else**, etc. are used.

Structured programming also encourages the decomposition of the code into small subprograms (called functions or procedures), which perform specific and independent tasks. These subprograms should avoid using global variables, which can cause unwanted side effects. They should rather use local variables and parameters, which are passed by value or by reference .

These techniques allow to create code that is easier to understand, test and modify. They also facilitate the design by successive decompositions (top-down design), which consists of starting from the general problem and dividing it into simpler subproblems, until reaching the level of elementary instructions.

Here is an example of content that you could use to illustrate the concept of structured programming to your students:

**Example:**

We want to write a program that calculates the sum of the first n natural integers, where n is entered by the user.

**Solution:**

We can use structured programming to solve this problem. We start by defining the general problem:

* Input: a natural integer n
* Output: the sum of the first n natural integers
* Processing: calculate the sum using a loop

We can then decompose the problem into simpler subproblems:

* Input: read n
* Output: display the sum
* Processing: initialize the sum to 0, initialize a counter to 1, repeat as long as the counter is less than or equal to n: add the counter to the sum, increment the counter

We can finally translate these subproblems into instructions in a programming language. For example, in pseudo-code:

// Read n

Read n

// Initialize the sum to 0

sum <- 0

// Initialize the counter to 1

counter <- 1

// Repeat as long as the counter is less than or equal to n

While counter <= n do

// Add the counter to the sum

sum <- sum + counter

// Increment the counter

counter <- counter + 1

End While

// Display the sum

Display sum

This program is structured according to the three principles of structured programming: sequence, alternative and repetition. It does not contain any goto instruction. It uses local variables (sum and counter) and a parameter (n). It could be encapsulated in a function or a procedure.

I hope this example was helpful to you. If you have any other questions about structured programming, feel free to ask me. 😊