|  |  |  |
| --- | --- | --- |
|  | C++ | Java |
| 1. Biến và hằng | 1. Biến:  - Cú pháp khai báo: kiểu dữ liệu <tên biến>;  - Có 5 loại:   * Biến cục bộ: được định nghĩa và sử dụng chỉ trong 1 hàm hoặc block, không được tự động khởi tạo giá trị. * Biến toàn cục: được định nghĩa bên ngoài các hàm, không thay đổi giá trị trong suốt chương trình, tự động khởi tạo giá trị. * Biến tĩnh: được khai báo với từ khóa ‘static’, có phạm vi trong một lớp và được chia sẻ giữa các đối tượng của lớp đó. * Biến auto: được khai báo với từ khóa ‘auto’, tự động xác định kiểu dữ liệu của biến dựa trên giá trị khởi tạo. * Biến extern: được khai báo với từ khóa ‘extern’, cho phép sử dụng ở các file mã nguồn khác nhau.   -Hỗ trợ con trỏ và tham chiếu, không có bộ thu gom rác tự động nên phải sử dụng ‘new’ và ‘delete để cấp phát và thu hồi bộ nhớ.  2. Hằng:  Có 2 cách định nghĩa:  - Sử dụng #define: #define tên\_hằng giá\_trị  - Sử dụng ‘const’: const kiểu\_dữ\_liệu tên\_hằng = giá\_trị; | 1. Biến:  - Cơ bản giống C++ về cú pháp khai báo.  - Có 3 loại:   * biến cục bộ: có phạm vi trong 1 phương thức, chỉ cho phép sử dụng và truy cập trong phương thức đó. * biến thành viên: có phạm vi trong một lớp và thuộc về tất cả các đối tượng thuộc lớp đó. * biến tĩnh: giống C++.   Không có biến toàn cục vì Java là ngôn ngữ hướng đối tượng hoàn toàn.  - Không hỗ trợ con trỏ và tham chiếu như C++.  - Java có bộ thu gom rác tự động (garbage collector) nên không cần lo lắng về việc cấp phát hoặc giải phóng bộ nhớ thủ công.  2 . Hằng:  Trong Java, có thể sử dụng từ khóa ‘final’ để đánh dấu một biến là hằng. Biến hằng chỉ có thể gán giá trị một lần, và sau đó không thể thay đổi giá trị đó. |
| 1. Kiểu dữ liệu | Gồm các kiểu dữ liệu cơ sở và các kiểu dữ liệu tự định nghĩa.  1. Các kiểu dữ liệu cơ sở:  • Kiểu số nguyên: gồm short, unsigned short, int, unsigned, long, unsigned long, long long, unsigned long long với phạm vi giá trị và kích thước bộ nhớ khác nhau  • Kiểu số thực: float, double, long double. Được viết dưới dạng thập phân hoặc khoa học nếu độ dài quá lớn.  • Kiểu luận lý/logic: bool. Gồm 2 giá trị true(1) và false(0).  • Kiểu void: kiểu dữ liệu rỗng không chứa gì cả.  • Kiểu ký tự: char, unsigned char. Biểu diễn thông qua bảng mã ASCII.  • Kiểu Typedef: từ khóa typedef dùng để đặt tên mới cho một kiểu dữ liệu có sẵn.  • Kiểu Enum: kiểu dữ liệu giúp hỗ trợ định nghĩa những giá trị liệt kê.  2. Các kiểu dữ liệu tự định nghĩa: sử dụng từ khóa ‘struct’ hoặc ‘class’ để tự định nghĩa một kiểu dữ liệu mới. | - Trong Java có 2 kiểu dữ liệu: primitive types và reference types  1. Primitive types: Gồm boolean, byte, char, short, int long, float và double   * Kiểu luận lý/logic là boolean thay vì bool. * Không có kiểu dữ liệu không dấu (unsigned) và long double. * Kích thước cho các kiểu dữ liệu này là cố định, khác với C++.   Các kiểu dữ liệu primitive không tham chiếu tới đối tượng nên không thể gọi phương thức.  2 . Reference types:   * Chương trình Java sử dụng các biến có kiểu reference (tham chiếu) để lưu trữ vị trí của các đối tượng trong bộ nhớ máy tính. Các biến như vậy tham chiếu tới các đối tượng trong chương trình, cho phép truy cập thuộc tính, gọi phương thức của đối tượng đó. Một số kiểu dữ liệu reference phổ biến: String, Array, int[], enum,... * Các biến reference được khởi tạo giá trị mặc định là null. |
| 1. Toán tử, biểu thức | 1. Có 9 loại toán tử:  • Gán(=): dùng để gán giá trị cho một biến  • Toán học(+, -, \*, /, %): tính các phép toán đại số  • Tăng giảm(++, --): đặt sau 1 biến để tăng hoặc giảm giá trị 1 đơn vị.  • Phẩy(,): các biểu thức đặt cách nhau bằng dấu ‘,’, các biể thức con lần lượt được tính từ trái sang phải, biểu thức mới nhận được là giá trị của biểu thức bên phải cùng.  • Bit (& “AND”, ^ “OR”, | “XOR”, ~ “NOT”, >> “DỊCH PHẢI”, << “DỊCH TRÁI”): tính toán hệ nhị phân, có thể dùng để kiểm tra chia hết, tích tích và thương.  • Toán học/bit kết hợp gán (-=, +=,..., >>=, &=,...): dùng giá trị của biến để tính toán và gán lại cho nó.  • Điều kiện (<điều\_kiện>?<biểu\_thức\_1>:<biểu\_thức\_2>): nếu điều kiện đúng thì thực hiện biểu thức 1 còn không thì thực hiện biểu thức 2.  • Quan hệ (>, <, >=, <=, ==, !=): làm phép toán so sánh và trả về true/false.  • Luận lý (! “NOT”, && “AND”, || “OR”): tính toán giữa các giá trị luận lý và trả về giá trị luận lý.  2. Biểu thức:  - Tạo thành từ các toán tử và các toán hạng. Toán hạng có thể là hằng, biến, lời gọi hàm,... |  |
| 1. Lệnh điều khiển | 1. Cấu trúc rẽ nhánh:  if (điều kiện) {  khối lệnh 1  }  else {  khối lệnh 2  }  HOẶC  switch (biến) {  case giá\_trị\_1: {khối lệnh 1}  case giá\_trị\_2: {khối lệnh 2}  ...  }  2. Cấu trúc lặp  for (Init; Expression; PostExpression) {  khối lệnh  }  HOẶC  while (expression) {  action  }  HOẶC  do {  action  } while (expression); |  |