1. Ondalık değer hafıza alanını nasıl boşa harcar?

Bazı eski diller, ondalık sayıları saklamak için kullanılan veri türlerinde dönüşüm kayıplarına neden olabilir. Örneğin, bir dilde bir ondalık sayıyı saklamak için kullanılan bir veri türü, belirli bir hassasiyeti koruyamayabilir ve bu da bilgi kaybına yol açabilir.

Eski düşük seviyeli dillerde, bellek yönetimi genellikle daha el ile yapılır ve programcının bellek kullanımını optimize etmek için daha fazla çaba harcaması gerekir. Yanlış bellek tahsisi veya serbest bırakma, bellek sızıntılarına ve dolayısıyla gereksiz bellek kullanımına neden olabilir.

1. C ve C++'daki -> operatörünün bulunma gerekçesi nedir?

-> operatörü, bir işaretçi (pointer) ile erişilen bir yapı (struct) veya sınıf (class) üyesine erişmek için kullanılır. Bu operatör, bir işaretçinin gösterdiği nesnenin bir üye öğesine erişmek için kullanılır.

3. C++'ın numaralandırma türleri ile Java'nınkiler arasındaki farklar nelerdir?

C++'da numaralandırma türleri, tamsayı değerlerine atanmış sabit isimlerin bir listesini temsil eder. Bu değerlerin varsayılan olarak 0'dan başlayarak artan bir şekilde sıralanır. Ancak değerler manuel olarak atanabilir.

Java'da enum türleri, sabitlerin bir türdür. Her sabitin kendi türü vardır ve bunlar varsayılan olarak 0'dan başlayarak artan bir şekilde sıralanmış olarak gelirler. Ancak Java'da da her bir sabite bir değer atanabilir.

C++, işaretli (signed) ve işaretsiz (unsigned) tamsayı türlerine sahiptir. Java ise yalnızca işaretli tamsayı türlerini destekler.

C++:

int: işaretli tamsayı

unsigned int: işaretsiz tamsayı

Java:

int: işaretli tamsayı, Java'da yalnızca işaretli tamsayı türüdür.

Java'da enumlar bir sınıf olarak kabul edilir ve bu nedenle içlerinde metotlar ve diğer özellikler bulunabilir. Bu, Java'daki enumların daha esnek ve güçlü olmalarını sağlar.

C++'da enumlar sadece sabit değerlerin listesini temsil ederler, bu yüzden metotlar veya diğer sınıf özellikleri içeremezler.

4.sum1 ve sum2 değerlerini belirlemek için C'yi destekleyen bir sistemde aşağıdaki kodu çalıştırın ve sonuçları açıklayın.

Kodu bir C sisteminde çalıştırınca başarılı bir şekilde çalışıyor. Ekrana müdahale eden bir kod kullanmadığımız için program çıktısı oluşmaz.

sum1 hesaplanırken, (i / 2) ifadesi 10'un yarısı olan 5'i verir. fun(&i) çağrılır ve i'ye 4 eklenir, bu da i'yi 14 yapar. fun(&i) işlevi, i'yi 3 ile çarpar ve 1 çıkarır, sonuç 41 olur. sum1 değeri, 5 ile 41'in toplamı olan 46 olur.

sum2 hesaplanırken, fun(&j) çağrılır ve j'ye 4 eklenir, bu da j'yi 14 yapar. fun(&j) işlevi, j'yi 3 ile çarpar ve 1 çıkarır, sonuç 41 olur. Daha sonra (j / 2) ifadesi 14'ün yarısı olan 7'yi verir.

Yani, sum2 = 41 + 7 = 48 olur.

5. Bir önceki programı C++, Java ve C# programlama dilleri ile yeniden yazın, çalıştırın ve sonuçları karşılaştırın.

C++ a çevrilmiş hali:  
  
#include <iostream>

#include <ctime>

int fun(int &k) {

k += 4;

return 3 \* k - 1;

}

int main() {

int i = 10, j = 10, sum1, sum2;

sum1 = (i / 2) + fun(i);

sum2 = fun(j) + (j / 2);

}

Java ya çevrilmiş hali:

public class Main {

static int fun(int[] k) {

k[0] += 4;

return 3 \* k[0] - 1;

}

public static void main(String[] args) {

int i = 10, j = 10, sum1, sum2;

sum1 = (i / 2) + fun(new int[]{i});

sum2 = fun(new int[]{j}) + (j / 2);

System.out.println("sum1 = " + sum1);

System.out.println("sum2 = " + sum2);

}

}

C# a çevrilmiş hali:  
  
using System;

class Program

{

static int Fun(ref int k)

{

k += 4;

return 3 \* k - 1;

}

static void Main(string[] args)

{

int i = 10, j = 10, sum1, sum2;

sum1 = (i / 2) + Fun(ref i);

sum2 = Fun(ref j) + (j / 2);

Console.WriteLine("sum1 = " + sum1);

Console.WriteLine("sum2 = " + sum2);

}

}  
  
  
Kodların işlevinde bir değişiklik yoktur. Değişiklik sadece syntax tadır.