Prosedür Çağrıları ve Dönüşler Nedir?

Çalışma Kağıdı

Prosedür çağrıları, bir programın belirli bir görev için başka bir kod bloğunu çalıştırmasını ifade ederken, dönüşler ise bu görev tamamlandıktan sonra programın kaldığı yerden devam etmesini sağlar. Bu süreç genellikle yığın (stack) veri yapısı kullanılarak yönetilir.

Sorular

1. Prosedür çağrılarını ve dönüşlerini yönetmek için genellikle hangi veri yapısı kullanılır?
A) Kuyruk (Queue)
B) Yığın (Stack)
C) Bağlı Liste (Linked List)
D) Ağaç (Tree)

2. Bir fonksiyon çağrıldığında, programın kaldığı yerin bilgisi nereye kaydedilir?
A) Sabit Bellek
B) Yığın (Stack)
C) Yazmaçlar (Registers)
D) Önbellek (Cache)

3. İç içe geçmiş iki fonksiyon çağrısında, ilk çağrılan fonksiyon ne zaman çalışmasına devam eder?
A) İkinci fonksiyon çağrılmadan hemen önce
B) İkinci fonksiyon çağrıldıktan sonra
C) İkinci fonksiyon işini bitirip döndükten sonra
D) Program tamamen bittikten sonra

4. Basit bir toplama fonksiyonu çağrısı nasıl çalışır?

5. İç içe fonksiyon çağrıları nasıl yönetilir?

6. Tanımla: Prosedür Çağrısı Nedir?

7. Tanımla: Prosedür Dönüşü Nedir?

8. Tanımla: Yığın (Stack) Yapısı

Cevap Anahtarı

1. B) Yığın (Stack) — Yığın (Stack), LIFO prensibi sayesinde fonksiyon çağrılarının doğru sırayla yönetilmesini sağlar; en son çağrılan fonksiyon ilk döner.

2. B) Yığın (Stack) — Fonksiyonun dönüş adresi (programın kaldığı yer), çağrı yığınına (call stack) itilir, böylece fonksiyon bittiğinde oraya geri dönülebilir.

3. C) İkinci fonksiyon işini bitirip döndükten sonra — Yığın yapısı gereği, en son çağrılan fonksiyon (ikinci fonksiyon) işini bitirip döndükten sonra, kontrol ilk çağrılan fonksiyona geri verilir.

4. Ana programda `topla(a, b)` fonksiyonu çağrılır.,Fonksiyonun adresi ve dönüş adresi yığına itilir.,Fonksiyonun parametreleri (a ve b) yığına itilir.,Kontrol `topla` fonksiyonuna geçer.,Fonksiyon çalışır ve sonucu hesaplar.,Sonuç yığından alınır veya bir değişkene atanır.,Dönüş adresi yığından alınır ve kontrol oraya aktarılır.,Ana program kaldığı yerden devam eder.

5. İlk fonksiyon çağrıldığında, adresi ve dönüş adresi yığına itilir.,İkinci fonksiyon çağrıldığında, kendi adresi ve dönüş adresi (ilk fonksiyonun devam edeceği yer) yığına itilir.,Her çağrı, kendi yerel değişkenlerini ve dönüş adresini yığına ekler.,Fonksiyonlar tamamlandıkça, yığından kendi bilgileri çıkarılır ve ilgili yerlere dönülür.,Bu yığın yapısı sayesinde iç içe çağrılar doğru sırayla çözülür.

6. Bir programın, belirli bir işlemi gerçekleştirmek üzere başka bir kod bloğunu (fonksiyon veya prosedür) çalıştırmasıdır.

7. Çağrılan prosedürün işini bitirdikten sonra, programın kaldığı yerden devam etmesi için kontrolü çağıran yere geri döndürmesidir.

8. Prosedür çağrıları ve dönüşleri yönetmek için kullanılan, LIFO (Last-In, First-Out) prensibiyle çalışan bir veri yapısıdır.