C dilinde dizi kavramı

C dilinde **dizi (array)**, aynı türdeki birden fazla veriyi bir arada tutmak için kullanılan bir veri yapısıdır. Diziler, bellekte ardışık olarak saklanan birden fazla öğeden oluşur ve bu öğelere indeksler aracılığıyla erişilir. Bir dizi, belirli bir türde (örneğin, int, float, char) sabit bir boyutta tanımlanır ve her bir öğe, dizideki konumuna göre bir indeks numarası ile tanımlanır. Diziler, programlama dilinde verilerin toplu olarak yönetilmesini sağlar ve işlemler üzerinde kolaylık sağlar.

Örneğin, 10 elemandan oluşan bir dizi tanımlandığında, bu dizinin her bir elemanı ardışık bellek hücrelerinde saklanır. Böyle dizi kullanmak yerine, 10 farklı isimle ayrı ayrı değişkenler tanımlamak da mümkündür. Ancak, bu durumda değişkenlerin bellekte ardışık bir şekilde yerleşeceği garanti edilemez. Bellekte rastgele yerleşen ayrı değişkenlerin yönetimi zorlaşabilir ve verilerin erişim süresi uzayabilir. Buna karşın, bir dizi tanımlandığında, dizinin her bir elemanı bellekte sıralı bir şekilde depolanır. Bu ardışık yapı, özellikle diziler üzerinde döngülerle işlem yaparken büyük avantaj sağlar ve belleğin verimli kullanılmasını garanti eder.

Dizilerin Özellikleri

1. **Sabit boyut:** Diziler tanımlandıklarında belirli bir boyuta sahip olurlar ve bu boyut programın çalışma süresi boyunca sabit kalır. Dizi boyutu, bellekte ayrılan yer miktarını belirler.
2. **Aynı türde elemanlar:** Bir dizi, yalnızca belirli bir veri türünden öğeler içerebilir. Örneğin, int türünde bir dizi sadece tamsayılar içerir.
3. **Sıralı bellek:** Dizideki öğeler, bellekte sıralı bir şekilde saklanır. Bu, her öğeye doğrudan indeksle erişimi hızlı ve verimli hale getirir.
4. **Sıfır tabanlı indeksleme:** C dilinde diziler, 0'dan başlayan indekslerle erişilir. İlk öğenin indeksi 0, ikinci öğenin indeksi 1 vb. şeklindedir.
5. **Doğrudan erişim:** Bir dizi elemanına doğrudan indeks numarası ile erişilebilir. Bu, belirli bir öğeye hızlı erişim sağlar.

Dizgilerin tanımlanması

Metin dizileri, dizgiler iki tırnak simgeleri (" ") arasında tanımlanırlar. Bu tanımlama şekli metin dizilerinin 1'den fazla karakterden oluştuğunu belirtmektedir. Daha önceki derslerde anlatıldığı üzere C dilinde tek harf metin değişkenleri tek tırnakla (' ') ayrı bir şekilde tanımlanmaktadır. Bu ise iki farklı şekilde karakter değişken tanımlanmasına neden olmaktadır. Bu açıdan kullanım amacına uygun olarak metin değişkenleri tanımlamak gereklidir. Metin dizileri 1'den fazla karakterden oluştukları için sayısal dizilerde olduğu gibi bir dizi formatından tanımlanmaları lazımdır.

Bir metin dizisinin tanımlama formatı/düzeni şu şekildedir:

```
char  dizgi ismi  [  uzunluk  ];
```

Dizgiler tanımlanırken bellek kullanımına dikkat edilmesi, taşmaların ve bellek hatalarının önlenmesi açısından önemlidir. Gereğinden fazla veya yetersiz bellek tahsisi, programın işleyişini olumsuz etkileyebilir. Kullanıcıdan alınacak girdilerin saklanması ve işlenmesi için ise dizgiler önemli bir araçtır. Tanımlama sırasında dizgi uzunluğunu ve formatını doğru şekilde belirlemek, kullanıcı girdilerinin sorunsuz bir şekilde işlemek için gereklidir.

1. **Sabit Boyutlu Tanımlama:** Sabit boyutlu diziler, belirli bir karakter sayısına uygun olacak şekilde tanımlanır. char dizi[20]; // 20 karakterlik bir yer ayrılır.
2. **Başlangıç Değeri ile Tanımlama:** Tanımlama sırasında bir metin atanır ve '\0' karakteri otomatik olarak eklenir. char dizi[] = "Merhaba"; // 8 karakterlik bir dizi (7 harf + '\0') oluşur.
3. **Karakter Dizisi ile Tanımlama:** Her karakter ayrı ayrı belirtilerek dizi tanımlanabilir. char dizi[5] = {'H', 'e', 'l', 'l', 'o'};
4. **Dinamik Bellek Tahsisi ile Tanımlama:** Daha esnek bir yapı sağlamak için malloc() veya calloc() kullanılarak bellekte yer ayrılır. char *dizi = (char *)malloc(50 * sizeof(char)); // 50 karakterlik bir alan oluşturulur.

Göstergeler/işaretçiler - pointers

C dilinde göstergeler (pointers), bir değişkenin bellekteki adresini saklayan özel türde değişkenlerdir. Göstergelerin temel işlevi, bellekteki verilere doğrudan erişim sağlamaktır. Bu özellik, programların çalışma şeklinde esneklik sunar ve aynı zamanda performansı artırır. Göstergeler, bellek yönetimi gibi düşük seviyeli işlemlerden, diziler ve fonksiyonlarla çalışmaya kadar birçok önemli alanda kullanılır. Örneğin, bir değişkenin değerine dolaylı yoldan erişmek veya bir dizinin elemanları üzerinde işlem yapmak için göstergelerden faydalanılır. Bu yönüyle göstergeler, programcıya hem güçlü bir kontrol mekanizması hem de etkili bir çözüm sunar.

Göstergeler, özellikle dinamik bellek tahsisi, veri yapılarının uygulanması ve işlevlere veri geçişinde kritik bir rol oynar. Dinamik bellek tahsisinde, göstergelerle bellekte ihtiyaç duyulan alan rezerve edilir ve işlem tamamlandığında bu alan serbest bırakılır. Ayrıca bağlantılı listeler, ağaç yapıları ve grafikler gibi karmaşık veri yapılarının oluşturulması ve yönetilmesi de göstergeler sayesinde mümkün hale gelir.. Bununla birlikte, göstergelerle çalışırken dikkatli olunması gerekir; yanlış bellek erişimi veya bellek sızıntısı gibi sorunlar programın çalışmasını olumsuz etkileyebilir.

Göstergelerin Genel Özellikleri

1. **Bellek Adreslerini Saklarlar:** Göstergeler bir değişkenin bellek adresini saklar, bu adres sayesinde değişkenin içeriğine erişilir.
2. **Doğrudan Bellek Erişimi:** Bellekte doğrudan erişim ve işlem yapma imkanı sunar.
3. **Dinamik Bellek Yönetimi:** malloc, calloc, ve free gibi fonksiyonlarla dinamik bellek tahsisinde kullanılır.
4. **Esneklik Sağlar:** Diziler, dizilerden oluşan diziler (multi-dimensional arrays), işlevler (functions) ve veri yapıları üzerinde işlem yaparken esneklik sağlar.
5. **Performansı Artırır:** Kopyalama yerine bellekteki veriyle doğrudan çalışıldığı için performans açısından avantaj sağlar.

Dosya İşlemleri

Dosya işlemleri dosyanın içeriği üzerinde olduğu gibi bulunduğu yer ve özellikleriyle ilgili işlemler de olabilmektedir. Dosya içeriği ile ilgili işlemler veri okumak, yazmak, düzenlemek ve yönetmek gibi çeşitli adımları içerir. Yer ve özellikleriyle ilgili işlemler ise dosya silme, kopyalama, taşıma ve erişim izinleri gibi adımlar sayılabilir.

Dosya içeriği üzerine yapılan işlemler:

Dosya Oluşturma: Yeni bir dosya oluşturulması, genellikle veri kaydedilecek bir alan sağlamak için yapılır. Bu işlemde dosyanın adı ve türü belirlenir.

Dosya Açma: Dosya açma işlemi, dosya üzerinde işlem yapmaya başlamak için yapılır. Dosya, okuma (giriş), yazma (çıkış) veya ekleme (append) gibi çeşitli modlarda açılabilir.

Dosya Kapatma: Dosya kapama işlemi, dosya ile yapılan tüm işlemler tamamlandıktan sonra yapılır. Dosya kapatıldığında, işletim sistemi dosyanın kaynaklarını serbest bırakır ve değişiklikler kaydedilir. Bu işlem, dosya işlemlerinin sonlandırılması için gereklidir.

Dosya Okuma: Dosya okuma, bir dosyanın içeriğinin program tarafından alınması işlemidir. Bu işlemde, dosya belirli bir formatta okunur ve içerik işlenmek üzere belleğe aktarılır.

Dosya Yazma: Dosya yazma, yeni veri eklemek veya mevcut veriyi değiştirmek amacıyla yapılan işlemdir. Yazma işlemi, dosyanın üzerine yazmak veya dosyaya yeni veri eklemek şeklinde olabilir.

Dosya yer ve özellikleri üzerine yapılan işlemler:

Dosya Silme: Dosya silme işlemi, bir dosyanın sistemden tamamen kaldırılmasını sağlar. Bu işlem, artık kullanılmayan veya geçersiz dosyaları sistemden temizlemek için yapılır.

Dosya Kopyalama ve Taşıma: Dosya kopyalama, bir dosyanın içeriğini başka bir yere çoğaltarak bir kopyasını oluşturmak anlamına gelir. Taşıma ise, dosyayı bir konumdan başka bir konuma fiziksel olarak yer değiştirme işlemidir.

Dosya İzinlerini Değiştirme: Dosya izinlerini değiştirme, dosyanın kimler tarafından okunabileceği, yazılabileceği veya çalıştırılabileceği gibi erişim haklarının belirlenmesi işlemidir ve güvenlik için önemlidir.

Standart (Hazır) Kütüphane

Bilindiği üzere **ANSI C**, C programlama dili için uluslararası bir standart olarak kabul edilir ve dile ait temel kuralları ve yapıları belirler. Turbo C, Microsoft C ve UNIX tabanlı C derleyicileri gibi çeşitli derleyiciler ANSI C standardını destekleyerek kullanıcıların yazdığı kodların farklı platformlarda sorunsuz çalışmasını mümkün kılar.

Her C derleyicisi, temel standartların yanı sıra kendi özel özelliklerini ve kütüphanelerini sunar. Bu ek özellikler, belirli platformlara özgü işlevsellik veya kullanım kolaylığı sağlamak için tasarlanmıştır. Örneğin, ANSI C'nin standart kütüphaneleri grafik fonksiyonlarını içermez; bu tür özellikler dilin kapsamı dışında bırakılmıştır. Ancak Turbo C, kullanıcılarına grafik fonksiyonları içeren özel bir kütüphane sunar ve bu, grafiksel uygulamalar geliştirmek isteyen programcılar için büyük bir avantaj sağlar.

ANSI C standart kütüphanelerini temel fonksiyonlar olarak aşağıda gösterilen şekilde gruplamak mümkündür:

- Standart Giriş/Çıkış fonksiyonları
- Matematiksel fonksiyonlar
- Zaman ve Tarih fonksiyonları
- Karakterler üzerinde işlem yapan fonksiyonlar
- Metin dizileri üzerinde işlem yapan fonksiyonlar
- Genel amaçlı fonksiyonlar

Standart kütüphanede bu fonksiyonların dışında başka işlemler için de fonksiyonlar bulunmaktadır. Bunlar ise şu başlıklar altında verilebilir.

- Esnek argüman aktarım fonksiyonları
- Yerel olmayan Atlama (Goto) fonksiyonlar
- Ayrıcalıklı Durum (Exceptional handler) Kontrolü ile ilgili komutlar
- Hata ayıklama (Diagnostics) fonksiyonları
- Yerelleştirme (Localization) fonksiyonları

Dinamik Bellek Yönetimi

C dilinde dinamik bellek yönetimi, bir programın çalışma zamanında ihtiyaç duyduğu kadar bellek ayırmasını ve artık kullanılmayan belleği serbest bırakmasını sağlayan bir sistemdir. Bu mekanizma, programın başlangıcında bellek boyutunun sabit olmadığı durumlarda veya büyük ve esnek veri yapılarına ihtiyaç duyulduğunda oldukça önemlidir. Örneğin, kullanıcıdan alınan verilere bağlı olarak değişen boyutlarda bir dizi oluşturmak veya programın çalışması sırasında yeni veri yapıları eklemek gerektiğinde dinamik bellek yönetimi devreye girer.

Dinamik bellek yönetime ait işlemler **stdlib.h** başlık dosyasında tanımlı fonksiyonlar kullanılarak gerçekleştirilir. Bu fonksiyonlar sayesinde bellek kullanımını optimize etmek, programın esnekliğini artırmak ve gereksiz bellek tüketimini önlemek mümkün olur. Dinamik bellek yönetimi, özellikle karmaşık ve uzun süreli çalışan uygulamalarda verimli bir şekilde kullanıldığında büyük bir avantaj sağlar.

Dinamik Bellek Yönetiminin Temel Kavramları

C dilinde bellek yönetimi, işletim sistemi tarafından sağlanan bir bellek havuzundan (heap) yapılır. Bu havuz, program çalışırken dinamik olarak genişletilebilir veya daraltılabilir. Bellek yönetiminde 3 temel kavram vardır ve bunlar aşağıdaki şekilde özetlenebilir.

1. **Bellek Tahsisi (Memory Allocation)**
   Bellek tahsisi, çalışma zamanında belirli bir miktarda belleğin program tarafından kullanılmak üzere ayrılmasıdır. Bellek tahsisi için kullanılan fonksiyonlar **malloc** ve **calloc** fonksiyonlarıdır.

2. **Belleği Yeniden Boyutlandırma (Reallocation)**
   Tahsis edilen bir belleğin boyutunu arttırılması veya azaltılması işlemidir. Bunun için **realloc** fonksiyonu kullanılır.

3. **Belleğin Serbest Bırakılması (Memory Deallocation)**
   Kullanılmayan bir bellek bloğunun serbest bırakması işlemidir, böylece sistemde başka işlemler için serbest bırakılan bellek kullanılabilir duruma gelir. Bu işlem için **free** fonksiyonu kullanılır.