A.Ü. GAMA MYO. Elektrik ve Enerji Bölümü

ALGORTİMA VE PROGRAMLAMA

4. HAFTA

İçindekiler

AKIŞ DİYAGRAMLARI

Herhangi bir sorunun çözümü için izlenmesi gerekli olan aritmetik ve mantıksal adımların söz veya yazı ile anlatıldığı algoritmanın,görsel olarak simge ya da sembollerle ifade edilmiş şekline "akış şemaları" veya FLOWCHART adı verilir.Akış şemalarının algoritmadan farkı,adımların simgeler şeklinde kutular içine yazılmış olması ve adımlar arasındaki ilişkilerin ve yönünün oklar ile gösterilmesidir.

Programın saklanacak esas belgeleri olan akış şemalarının hazırlanmasına,sorun çözümlenmesi sürecinin daha kolay anlaşılır biçime getirilmesi, iş akışının kontrol edilmesi ve programın kodlanmasının kolaylaştırılması gibi nedenlerle başvurulur.Uygulamada çoğunlukla, yazılacak programlar için önce programın ana adımlarını(bölümlerini) gösteren genel bir bakış akış şeması hazırlanır.Daha sonra her adım için ayrıntılı akış şemalarının çizimi yapılır.

AKIŞ DİYAGRAMLARI

Akış şemalarının hazırlanmasında yer alan simgelerden bazıları şunlardır.

1. 

Algoritmanın başladığını ya da sona erdiğini belitmek içinkullanılır.

2. 

Klavye aracılığı ile giriş ya da okuma yapılacağını gösterir.

3. 

Yazıcı(printer) aracılığı ile çıkış yapılacağını gösterir.

SABİTLER

Sabitler program boyunca değeri değiştirilemeyen nesnelerdir. Sabitler birçok durumda derleyici programın parçası gibi davranırlar ve RAM'den ziyade ROM da saklanırlar.

X+5 atamasında *"5"* bir sabittir ve derleyici tarafından doğrudan toplama işlemine sokulur.

printf ("selam"); ifadesinde *"selam"* metni program belleğine yerleştirilir ve değeri değiştirilemez.

Programlayıcıda const özel sözcüğünü kullanıp sabit bildirimi yapabilir. Sabit bildiriminde const özel sözcüğü ile birlikte tanımlayıcı ismi, türü ve değeri belirtilmelidir.

constant unsigned char k =80;

Bir nesneyi sabit olarak tanımlamak nesnenin kısıtlı depolama alanı bulunan RAM yerine program kodunda saklanmasını neden olur. Bu ise sınırlı RAM alanının korunmasına yardımcı olur.

DEĞİŞKENLER

a) Değişken Bildirimi

Bir değişkenin programda kullanılabilmesi için program içerisinde bildiriminin yapılması gerekir. Değişkenin bildirimi, değişkenin türünü ve boyutunu gösteren özel amaçlı bir sözcük ve bir tanımlayıcı ile yapılır.

b) Değişken Kapsamı

Değişkenlerin programda kullanılabilmesi için programın başında bildirimlerinin yapılması gerekir. Bir değişkenin kapsamı program içerisindeki erişilebilirliğidir. Değişken program içerisinde yerel yada genel olarak tanımlanabilir.

DEĞİŞKENLER

- **Yerel Değişkenler:** fonksiyon tanımlandığı zaman, fonksiyon tarafından ayrılan bellek alanıdır. Bu değişkenler sadece tanımlandığı fonksiyon içerisinde anlamlıdır ve diğer fonksiyonlar tarafından kullanılamaz.

Derleyici yerel değişkeni yalnızca tanımlandığı fonksiyonun bir parçası olarak gördüğü için, farklı fonksiyonlarda aynı isme sahip yerel değişkenler tanımlanabilir.

- **Genel Değişkenler:** Derleyici tarafından ayrılan bellek alanları olup, bütün fonksiyonlar tarafından kullanılabilir.

DEĞİŞKENLER

Örnek:

```
#include <stdio.h>
int sayi;//Genel değişken
int main()
{
	int sayi_mod_2;//Yerel değişken
	int sayi_mod_4;// Yerel değişken
	sayi=1000;
	sayi_mod_2 = sayi >> 1;
	sayi_mod_4= sayi >>2;
	printf("mod_2   = %d\n\r mod_4    = %d",sayi_mod_2,sayi_mod_4);
return 0;
}
```

ATAMA İŞLEMLERİ

= operatörü: Basit eşitleme işlemlerinde kullanılır.

+= operatörü :Eşitliğin sağıdaki sayıyla kendisini toplayarak kendine eşitler.

++ değişkenden sonra yapılırsa önce atama işlemi yapılır sonra artırma yapılır.

++ değişkenden önce kullanılırsa önce artırım yapılır daha sonra atama işlemi yapılır.

-= operatörü :Eşitliğin sağıdaki sayıdan kendisini eksilterek kendine eşitler.

— değişkenden sonra yapılırsa önce atama işlemi yapılır sonra azaltma yapılır.

— değişkenden önce kullanılırsa önce azaltma yapılır daha sonra atama işlemi yapılır.

***= operatörü :** Eşitliğin sağıdaki sayıyla kendisini çarparak kendine eşitler.

/= operatörü : Eşitliğin sağıdaki sayıya kendisini bölerek kendine eşitler.

VERİ TİPLERİ

C dilinde veri tipleri değer ve referans olmak üzere 2'ye ayrılır.

Değer tipleri belleğin stack alanını kullanırken, referans tipleri heap bellek bölgesinde tutulur.

Değer tiplerinde tutulan verilere direk ulaşılabilirken, heap bölgesinde ise verilere ulaşmak için verinin adres bilgisini içeren bir referans tutulur ve veriye dolaylı bir erişim sağlanır.

Kaynakça

http://www.yildiz.edu.tr/~wwwhid/TR/algoritma3.htm

http://www.fpganedir.com/embedded/c_yapisi/degisken.php

http://www.gorselprogramlama.com/atama-islemi-c-console-programlama-temelleri-ders-6/

https://www.kodlamamerkezi.com/c-net/csharp-degisken-tanimlama-ve-veri-tipleri/