NECATİ ERGİN

C PRAGRAMLAMA EĞİTİM NOTLARI

1.DERS

C DİLİNİ TANIMA

C dilinin kökeni algol dilidir.

Procedural programlama dilidir.

C non-proprietary dildir.(birine ait olmayan)

Program son olarak işlemcinin komutlarında çalıştırılır.

Derleyiciler kodu makine diline çevirir.

Derleyici daha yüksek seviyeli dilden daha düşük seviyeli dile dönüştüren araçlardır.

Statik türde bir dildir.

C taşınabilir bir dildir.

Çevirici yani derleyici ile istediğimiz sisteme uygun programlama yapılabilir.

C ve c++ verim kritik projelerde kullanılır.

//

2.DERS

Source file (kaynak dosyası)

↓

Ön işlemci programı

↓

Translation unit

↓

Derleyici

↓

Hedef kod(object kod)

↓

Linker(bağlayıcı)

↓

Çalıştırılabilir kod

Kodların yazıldığı yer txt editörüdür.

Bilgisayarda hafızada dosyalar bit bayt şeklinde tutulur.

Derleyiciden çıkan assemby kodları assember ile makine kodlarına dönüştürülür.

Derleyici çıktısına object code denir.

Derleyicinin oluşturduğu hedef kodu linker yani bağlayıcılar özel bir yapıyla birleştirip çalıştırılabilir bir kod üretir.

Derleyici kaynak dosyaları tek tek derler.

Linker farklı kaynak dosyalarındaki kodları birbirleriyle ilişkilendirip uyumlu çalışmasını sağlar.

Compile time(derleme zamanı)

Link time (bağlama zamanı)

Run time (çalışma zamanı)

C standartında kaynak dosya ön işlemcinin girişidir.

Kaynak kodun ön işlemciden çıktığı koda translation unit denir.

Derleyicinin girişi translation unit tir.

Kodun yazıldığı yer derleyici değil editördür.

Derleyici çeviricidir.

Program çalıştırıp hataları görmemizi sağlayan bu yaparken yardımcı araçlar kullanan programlara debugger denir.

Her dilin kuralları vardır. Bunlara syntax adı verilir.

Derleyici syntax kurallarına bakar. Dilin kurallarını çiğnediğimiz yerlerde derleyici ileti verir.

Bunlar error veya warning dir.

Warning 2 anlamda kullanılır.

```c
#include <stdio.h>

int ival ;
int main()
{
    ival = 10;
    ival == 10;

}
```

Burada dil kurallarına uygundur. Ancak dil uyarı mesajı verir.

```c
#include <stdio.h>

int ival ;
int main()
{
    ival += 10;
    ival + 10;

}
```

Burada da bir kurallarına uygundur ama ival değeri 10 artmayacaktır.

Derleyici burada uyarı mesajı verir.

Static code analyzer logic hataları gösterir.

Compiler optimization da derleyici kodun anlamını bozmadan daha kısa sürede çalışacak şekilde ayarlar.

!!!!!!!!!!TANIMSIZ DAVRANIŞ !!!!!!!!!!!!

Tanımsız davranışlar tehlikeli durumlardır.

```c
int main()
{

x=f1() + f2();

}

```

Burada f1 veya f2 fonksiyonları sırası karışık olarak çağırılabilir.

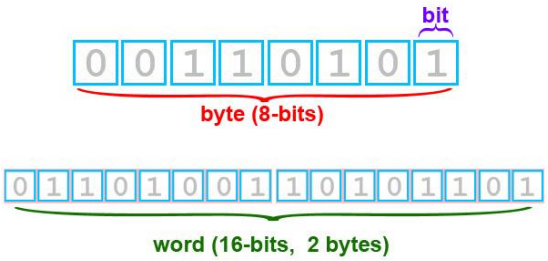
Hangisinin önce çağırılacağının bir garantisi yoktur. Bu belirsiz bir davranıştır.

//

3.DERS

SAYISAL SİSTEMLER

2lik sayı sistemi



4 bit = nibble

8 bit = byte

16 bit = Word

32 bit = Double Word

1100 1001 sayısının decimal değeri kaçtır ?

213

45 sayısının binary gösterimi nedir?

00101101

Binary değerlerin alabileceği maksimum değerler

1 byte =255

2byte =65.535

4 byte =4.294.967.295

8 byte = 18.446.744.073.709.551.615

Bir sayının 1 e tümleyeni

Sayının bitlerini terleyerek yapılır

11111111 →00000000

Bir sayının 2 ye tümleyeni

Sayının 1 e tümleyeninin 1 fazlasıdır. Kısa yol olarak sağdan başlayarak ilk 1 bitini görene kadar aynısını yazarız.1 bitini göründe ilk bit aynen yazılır geri kalanı ters çevrilir.

0111 0110 → 1000 1010

İşaretli 2lik sayı sistemi

8 bitlik sayının en son sol bitine işaret biti denir.

İşaret biti 0 ise pozitif

1 ise negatiftir.

Aynı sayının negatifi pozitif olan sayının 2 ye tümleyenidir.

0010 0100 = 36

1101 1100= -36

1111 1111 ?

Hepsi 1 ise -1 dir.

Hex sistemi 16lık sistemdir.

0111 1111 değeri kaçtır?

7F

2 byte sayının en küçük değeri

0111 1111 … 8000

2 byte alanda -1 değeri kaçtır?

FFFF

TERİMLER

Token = atom

Tokenazing derleyicinin komutları anlarken kullandığı yöntemdir.kodu token lara ayırır.

Translation unit ile source file arasındaki fark nedir?

Source file ön işlemci girdisidir.

Derleyicinin girdisi ise ön işlemci çıktısı translation unit tir.

Anahtar sözcük

Sabitler

İsimler

Operatörler

String literalleri

Ayraçlar

Anahtar sözcük (keyword)

İnt, if, for, do , float vb

Sabitler

41 decimal

0x41 hex

041 octal 8 lik sayı sistemi !!!!!!! **benzeri mükatta çıktı unutma**

C kurallarında olmayan bazı şeyler derleyicilerde farklılık gösterebilir.

Operatörler

Operatör işlem yaptıran tokendir.

String literalleri

Tek bir tokendir. “merhaba dünya”

object

Expression

Valve category

L valve

R valve

Constant

C de bellekte bir yere sahip bazı nitelikleri olan varlıklara nesne nedir.

Nesnenin türü nesnelerin ne olduklarını ifade eder.

Veri bellekte kaç byte yer kaplar?

1 ler ve 0 lar nasıl yorumlanacak?

Bu soruların cevaplarını nesnenin türü verir.

Veri türleri temelde 2 ye ayrılır.

Basic types ve user defined types

Basic types dil tarafında hazır sunulan tam sayı veya gerçek sayı değerlerini ifade eden türlerdir.

User defined types hazır türler dışında yeni oluşturduğumuz türlere verilen addır.

Expression(ifade)

Statement (deyim)

Declaration(bildirim)

Bir ismin ne olduğunu anlatan şeyler declaration dur.

İnt x ;

İnt foo(int x) ; gibi

Deyimler ise işlem kodu üreten şeylerdir.

X=y+5;

Value category işlemleri niteler

X+10

X*y+5 gibi

L-value

R-value

İfade bellekte bir yere karşılık geliyorsa l-value dur.

r-value bir hesaplama ifadesidir.

Hangi value kategorisine ait olduğunu anlamak için bir test yapabiliriz.

```
#include <stdio.h>


int main()
{

int x =10;
&(x);

}
```

Eğer syntax hatası vermiyorsa bir adresi vardır ve l-value dur.

Syntax hatası veriyorsa r-value dur.

Constant expression

İfadenin değerinin programın çalışma zamanı dışında derleyicide anlaşılan değerlere constant denir.

//

4.DERS

Derleyicinin isim arama sürecine name lookup denir.

İlk programı yazdıralım.

```c
#include <stdio.h>

int main (void)
{
printf("hello world");
}
```

Veri türleri (data types)

Her değişken verinin türü derleme aşamasında bilinmesi gerekir.

Basic types

User defined types

Basic types 2 kategoriye ayrılır.

İnteger types

Floating types

İnteger types

_Bool

Char

Signed char

Unsigned char

Short int

Unsigned short int

Long int

Unsigned long int

Long long int

Unsigned long long int

İnt

Unsigned int

Floating types

Float

Double

Long double

Değişken bildirimi

Global namespace

Local namespace

Global isim alanında statement yazılamaz sadece bildirim yapılabilir.

Yerel isim alaında ikiside yapılabilir.

İlk değer verme (initialization)

Değişkenin yaşam süresi

Lifespan

Storage duration

Global isim alanına yazılan ve yerel isim alanında static ön eki alan değişkenler statik değişken olurlar.

Değişken global ise ilk değer vermesekte 0 değerini alır.

Statik değişken olur ve bellekte kalır.

Değişkenin ömürüyle programın ömürü aynıdır.

Otomatik ömürlü değişkenler ise bulunduları scope kadar programda hayatta kalırlar.

Örnek

```c
#include <stdio.h>

void func(void)
{
    int x =10;
    printf("x=%d",x);
    x +=10;
}


int main (void)
{
 func();
 func();
 func();
 func();
 func();
}
```

Bu örnekte değişkenimiz otomatik ömürlü olduğu için her zaman 10 değerini verir.

```c
#include <stdio.h>

void func(void)
{
    static int x =10;
    printf("x=%d",x);
    x +=10;
}


int main (void)
{
 func();
 func();
 func();
 func();
 func();
}
```

Ancak burada statik ömürlü olduğu için her fonsiyonu çağırdığımızda 10 artarak gelir.

!!!!!!!!!!!!!!!!!!! ilk tanımsız değer örneği!!!!!!!!!!!!!!!!!!

Otomatik ömürlü değişkene ilk değer vermezsek değişken çöp değer ile hayata başlar.

Program ve çalışması için hiç istemeyen bir durumdur.

//

5.DERS

İnitialization is not a assignment

İlk değer verme bir atama değildir.

C dilinde statik ömürlü değişkenlerinin ilk değerleri constat olmadır.

Örnek

İnt x= 10 ; kurallara uygun

İnt y=x ; syntax hatası

Otomatik ömürlü değişkenlerde böyle bir zorunluluk yoktur.

Scope nedir

Scope / namespace

Soru bildirdiğimiz bir ismi hangi kaynak kod alanında kullanabilirim.

Sorunun cevabı scope dur

İsim ara işlemi cok önemlidir

isim arama billi bir sırayla yapılır.

Aranan isim bulununca isim arama işlemi biter.

Basit bir mülakat sorusu

```
int main (void)
{

 int printf =10;
 printf("hello world");
}
```

Burada isim arama sırasında printf i değişken olarak gösterdiğimiz için syntax hatası alıyoruz.

İsim aramayı kullanıldığı ve üstende kalan alanda yapar.

Soru Aynı ismi birden fazla varlığa verirsek ne olur?

Eğer 2 isimin kapsamı aynı ise birde fazla varlığa verilemez ancak kapsamları farklı ise syntax hatası olmaz. Buna karşın iyi bir kullanım yöntemi değildir.

Scope türleri

File scope

Block scope

File scope

Global isim alanında bildirildikleri noktadan dosyanın sonuna kadardır.

Block scope

Bildirildiği yerden bulunduğu bloğun sonuna kadardır.

Örnek

```
int main (void)
{

 int x;
 double x[];

}
```

Geçersiz kod

```
int main (void)
{
 int x;

 if()
 {
 double x[];
}
}
```

Geçerli kod

Scope ları farklı olduğu için geçerlidir.

İsmin maskelenmesi

Yerel isim global ismi maskelerse global değer yazırılamaz.

Örnek

```
#include <stdio.h>

int x=70;

int main (void)
{
 int x =10;

 printf("x=%d",x);


}
```

Yerel int x global x i maskelediği için çıktı 10 değerini alır.

!!!Kursta Necati Ergin global x değişkeni yazdıramayız dedi ve bu dikkatimi çekti x değişkeninin nasıl yazdırabileceğimi düşünerek deneme yaptım ve buldum.

```c
#include <stdio.h>

int x=70;
void func(void)
{
 printf("x=%d",x);
}
int main (void)
{
 int x =10;

 func();

}
```

Bu kodun çıktısı 70 olur.

Burada func fonksiyonunu çağırdığımız zaman x ismini aramaya func fonksiyonunun içinden ve üstünden başlar.

Önemli bir mülakat sorusu

Ekran çıktısı ne olur.

```c
#include <stdio.h>

int x=70;

int main (void)
{
 int x =x;
 printf("%d",x);
}
```

Burada tanımsız davranış örneği vardır.

x değerininin çöp değeri ile tekrar kendisine değer atanmıştır.

Global ve yerel statik değişkenlerin ilk değerleri constant (sabit) olmalı.

Static ön adı gelen kod satırındaki bütün değişkenler statik değişken olur.

Costant

İnt main()

Const int x=10;

Cons değişkeninin değeri hiçbir zaman değiştirilemeyeceğini gösterir.

Değişmeyen değişkenin yararları nelerdir ?

Lojik hataların önüne geçer.

Kodun niyetini daha iyi anlatır.

Optimizasyonun daha iyi olmasını sağlar.

Static ve const ön adları birlikte kullanılabilir.

Sabitler kodun bir parçasıdır ve bellekte yer kaplamazlar.

//

6.DERS

C büyük harf küçük harf duyarlı bir dildir.

Tamamen büyük harf ile isimlendirme yapmak iyi bişey değildir.

C de genelde makrolar büyük harflidir. Bu yüzden koda bakan birisi kafası karışık yanlış anlayabilir.

Global değişkenler dezavantajlıdır.

Programın taşınabilirliğini azaltır.

Hata bulmayı zorlaştırır.

Değişkenin scope nu mümkün olduğunca daraltılması gerekir.

Önemli

Statik ömürlü değişken ile otomatik ömürlü değişkenin bulunduğu bellek farklıdır.

Otomatik ömürlü değişken ana bellek (stack) da tutulur.

Statik ömürlü değişken ana bellek (data segmenti) inde tutulur.

Örnek

```c
#include <stdio.h>


int main (void)
{

 for(int i = 0; i < 10; i++)
 {

  int x=5;
  printf("%d",x);
  ++x;
}
}
```

Bu örnekte ekranda x değerini her zaman 5 olarak gösterir.

Çünkü her seferinde döngüye tekrar giriyor ve x değeri bellekten siliniyor.

FONKSİYONLAR

Client code

To define a function

To call a function

Client code bir fonsiyonu çalıştıracak verileri gönderen kod

To define a funcion bir fonksiyonu tanımlamak

To call a function fonksiyonu çağırmak

Soru

Fonksiyon kendini çağıran koda nasıl veri iletir?

1) Geri dönüş değeri (return value)
2) Call by reference (değişken adresini gönder fonksiyon değişkenine hesapladığı değeri yazsın)

Double get_area(double Radius) fonksiyon syntax ında içeride noktalı virgül olmaz.

Void func(int x, y , z) **syntax hatası**

Void func(int x . int y , int z) **doğru syntax**

Fonksiyonların geri dönüş değeri olmak zorunda değildir.

Geri dönüş değeri olmayan fonksiyonlar void ön adıyla yazılır.

Eğer fonksiyonu tanımlarken void yazmak yerine boş bırakırsanız int olarak algılanır.

örnek

func(int x)

return 1;

derleyici int yazdığımızı düşünür.

Normalde dilin kurallarında böyle bişey yoktur.

Void clear_screen(void)

Parametrenin içindeki void değişkenin başlangıç parametresinin olmadığını gösterir.

Boş bırakırsak paremetre olabilir veya olmaya bilir diye düşünür.

Void func (int x,int y)

X ye y argümandır.

Void func (int x …..) → variyadik fonsiyondur.

İstediğimiz kadar argüman yazabiliriz.

Fonksiyonun parametre değişkeni ile içindeki değişkenin farkı

Variyadik fonksiyon yazmak için en az bir tane paremetre yazmalıyız.

Parametreyi çağıran kod gönderir.

Statement

1) Expression statement
2) compound statement
3) null statement
4) control statement

expression statement

x=5;

++x;

Func();

Compound statement

{

x=5;

++x;

Func();

}

Blokların içine yazılmış.

Null statement

; tek şekilde noktalı virgül kullanılır.

Control statement

Önceden bilirlenmiş bir syntax a sahip ve bu syntax gereği en az bir keyword kullanılan yapılar.

Prgramın akısını değiştirebilir.

1) İf statement
2) Loop statement
3) While statement
4) Do while statement
5) For statement
6) Switch statement
7) Goto statement
8) Return statement
9) Break statement
10) Contiune statement

Void ve return anahtar kelimeler

Bir fonksiyonun geri dönüş değeri olacaksa bunu return ddeyimi ile yapar.

Return ; yalın return deyimi

Return expr ; ifadeli return deyimi

Soru

Neden 2 tene return deyimi var?

Return deyiminin 2 işlevi vardır.

Çalışmakta olan fonksiyonun çalışmasını durdurması

Geri dönüş değeri üretmesi

```
void func(void)
//statement/
//statement
    if ()
    return;
    //statement
    //statement
```

Normalde return deyimine ihtiyaç yoktu. Ancak programın istediğimiz bir yerde durması için return deyimini kullandık.

Void bir fonsiyonun sonuna return deyimi koyarsak syntax hatası olmaz. Ancak okuyanın yanlış anlamasını sağlayabilir. Gereksizdir.

İfadeli return deyimi

Geri dönüş değeri olan fonksiyonda ifadesiz return kullanmayın.

İfadeli return deyiminde return ifadenin değeri hesaplanır.

Geri dönüş değeri çağıran koda iletilir.

Fonksiyonun kodunun çalışması sona erer.

Soru

Fonksiyonda bir adet mi geri dönüş değeri vardır?

Fonksiyonda bir adet geri dönüş değeri vardır.

İfonksiyonun içerisinde birden fazla return deyimi bulunabilir. Yalnızca bir tanesi geri dönüş değeri olarak iletilir.

Soru

İki tam sayıyı toplayan fonksiyon yazınız

```
int func(int a , int b)

    return a+b;
```

→ one-liner func (tek satır fonksiyon)

Soru

İki sayının en büyüğünü bulan fonksiyon yazınız.

```
int func(int a , int b)

    return a > b ? a : b;
```

Soru

üç sayının en büyüğünü bulan fonksiyon yazınız.

```
int max3(int a , int b , int c)
{
    int max =a;

    if(b>max)
    max=b;

    if(c>max)
    max=c;

    return max ;

}
```