***Pointer notlar***

//

// main.c

// pointer giriş

//

// Created by Asude Ekiz on 28.10.2021.

//

#include <stdio.h>

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** \*ip, x; //int bir pointer ve degisken tanimladik

x=21; //x e deger atadik

ip=&x; // pointera da x degiskeninin adresini atadık

printf("%d\n", x); //x i direkt yazdirdik

printf("%d\n", \*ip); // \*ip yaparak pointerın tuttugu adresteki degeri, yani x i yazdirdik.

//calistirinca her ikisi de 21 olmali

//adreslere gecelim

printf("%p\n", &x);// &kullanarak x in adresini yazdiralim

printf("%p\n", ip); //pointerı kullanarak x in adresini yazdiralim

//isaretciler yoluyla degiskenlere deger atayabiliriz

**int** y=10; //baslangıcta boyle olsun

**int** \*ip2=&y;

printf("%d\n",y);//10

printf("%d\n",\*ip2);//10

printf("%p\n", &y);//y in adresi

printf("%p\n",ip2);// y in adresi

//y in adresindeki degeri pointer vasitasiyla degistirelim

\*ip2=15 ;//olsun (yani dedik ki y yi gosteren pointerin ici artık 10 degil 15 olsun.)

//simdi y i yazdiralim

printf("%d\n", y);

//ama y nin adresini degistirmedik

printf("%p\n",&y);

printf("%p\n",ip2);

printf("%d\n",\*&y);//y nin adresine git adresteki degeri yaz

printf("%d\n",\*ip2) ;//y nin pointerına git ordaki degeri yaz

**return** 0;

}

2.ders

#include <stdio.h>

**int** main(**int** argc, **const** **char** \* argv[]) {

//ONEMLİ NOT

// int \*ptr derkenki yildiz ile sonradan \*ptr yapman ayni sey degil

// int \*ptr yaparkenki yildiz sadece bi pointer olarak tanimladigimizi belirtiyo

/\*

//simdi once f tanimlayalim, dolayli olarak degerini belirleyelim sonra point edelim

int f;

\*&f=4; //f in degerini f in adresinin degeri,,, diyerek tanimladik dolayli yoldan yani

int \*ip4=&f;

printf("%d\n",f);//4

printf("%d\n",\*ip4);//4

printf("%p\n", &f);//f in adresi

printf("%p\n",ip4);// f in adresi

\*/

//burda da m tanimlayalim ama deger bildirmeden point edelim ve icerigini soyle diyelim?? nolcak

**int** m;

**int** n;

**int**\* ip4=&m;

**int**\* ip5=&n;

printf("%d\n",\*ip4);//0 yazdırır cunku icerigi bilmiyo ki dsjgjk

printf("%d\n",\*ip5);//yine 0 mesela

\*&m=5; //simdi dolayli yoldan m in degerini bildirdik

printf("%d\n",\*ip4);//burda 5 yazdirir artik cunku biliyo

printf("%d\n",m);//5

\*&n=6;// yine dolayli yoldan n i bildirdik

printf("%d\n",\*ip5);//burda 6 yazdirir artik cunku biliyo

printf("%d\n",n);//6

printf("%p\n", &m);//m in adresi

printf("%p\n",ip4);// m in adresi

**return** 0;

}

3.ders

#include <stdio.h>

**int** main(**int** argc, **const** **char** \* argv[]) {

//adresi direkt kendimiz belirleyelim

//16F8A olsun

**char** \*ptr= (**char** \*)0x16f8a;

**char** \*ptr2= (**char** \*)0x7ffeefbff43c;

printf("ptr: %p\n",ptr);

printf("ptr2: %p\n",ptr2);

//o kafamiza gore verdigimiz adrese atama yapalim ve o degeri yadirmaya calisalim bakalim erisimimizin

//oldugu bir adres miymis

\*ptr=20;

printf("\*ptr: %d\n",\*ptr); // hata veriyor demek ki ulasabilecegimiz bi adres degilmis

//daha once baska bi kodda x diyip onun adresini yazdirmistim o adresi atadigimiz ptr2 ye deger atayalim

//ulasilabilir bi adres oldugundan o degeri gosterecek

\*ptr2=30;

printf("\*ptr2: %d\n",\*ptr2);

**return** 0;

}

Pointerlar neden kullanılır?

* call by reference
* dizileri fonksiyona aktarmanın tek yolu
* bir fonksiyon birden fazla değer döndüremez. ancak pointerlar sayesinde birden fazla değer aktarmak mumkun
* storafe’ı buyuk olan türlere pointer parametre olmayan fonksiyonlara argüman geçmek maliyetli ve yavaş bir işlemdir. pointer parametre kullanılarak maliyet ve performans artışı sağlanabilir
* bellek bölgesine doğrudan erişim
* pointerların doğru kullanımıyla performans ve maliyet artışı sağlanabilir

#include <stdio.h>

**void** func(**int** a,**int** b){ //call by valueye ornek bi fonksiyon

**int** temp;

temp=a;

a=b;

b=temp;

}

**void** func2(**int** \*a, **int** \*b){ //cal by reference'e ornek bi fonksiyon

**int** temp;

temp=\*a;

\*a=\*b;

\*b=temp;

//fonksiyonun icine ADRES yollicaz

//sonra temp zaten int deger, pointer degil, ona \*a yaparak bir deger yollicaz cunku burda int \*a diyerek

// bir pointer tanimladik (sey gibi iste int \*ptr bi pointer, sonra \*ptr yaparsan degere ulasirsin)

//a adresinin degerini b adresinin degerine atadik sonra b adresinin degerine de tempi atadik swap gerceklesti

}

**int** main(**int** argc, **const** **char** \* argv[]) {

//call by value

**int** a=20;

**int** b=30;

printf("a= %d b=%d\n",a,b);

func(a, b);

printf("a= %d b=%d\n",a,b);

// a ile b nin degeri degismedi CUNKU o void fonksiyonunda kopyalari uzerinde swap islemi yapıldı.

//call by reference

func2(&a, &b);

//bu fonksiyona ADRESLERİ yolladik ve adreslerin degerini swapleyerek

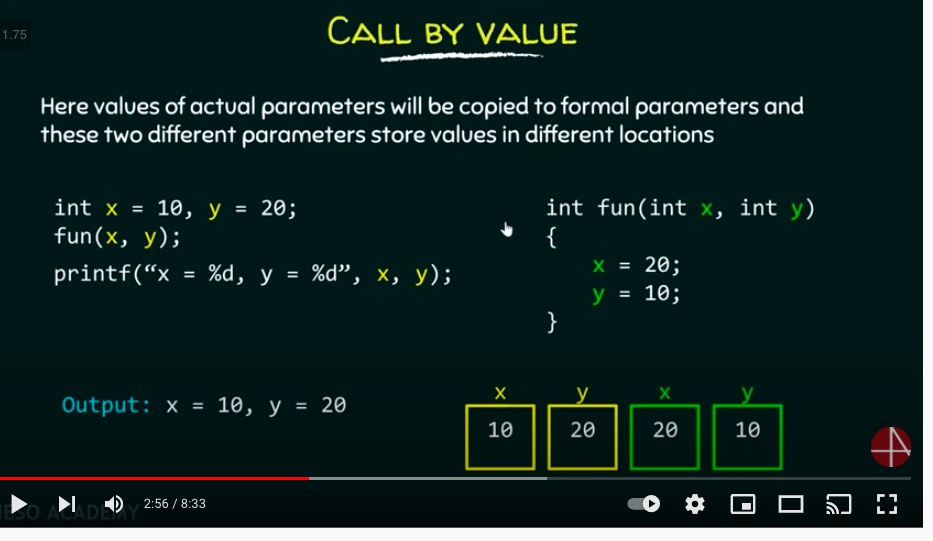
//kopyasi degil gercek degerler uzerinde oynamis olduk

printf("a= %d b=%d\n",a,b);

//burda artik swaplenmis halini yazicak

**return** 0;

}



metin, ekran, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

#include <stdio.h>

**int** fonk(**int** a,**int** b,**int** c){

**int** toplam;

toplam=a+b;

c=2\*c;

**return** toplam;

}

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** x=20;

**int** y=30;

**int** z=10;

printf("x: %d y: %d z:%d\n",x,y,z);

printf("toplam: %d\n", fonk(x, y, z));

printf("x: %d y: %d z:%d\n",x,y,z);

//FONKSİYON VE CALL BY VALUE'YA ORNEK

**return** 0;

}

NULL POINTER

Hiçbir nesneyi işaret etmeyen pointer

#define NULL ((char \* )0)

#define NULL 0

#define NULL 0L

üçü de aynı şey

#include <stdio.h>

**int** \*ptr2;

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** \*ptr=**NULL**;

printf("%p\n", ptr); //0 adresini gosterecek yani hicbir nesneyi isaret etmiyor.

printf("%p\n", ptr2); //bu da 0 adresini gosterecek cunku global isim alanında olusturdugumuz pointerlar da aslında null pointerdır.

**return** 0;

}

// printf("%d\n",\*ptr);

// bir yeri göstermeyen pointerin icindeki degeri ogrenmek istersek hata aliriz

// printf("%d\n",\*NULL); //bunda da ayni sekilde

//\*NULL=20;

//erisemedigimiz yere atama yapsak da yine hata

// printf("%d\n",\*NULL);

***Bazı pointer durumlari***

//bildiğim seylerdi

***Pointer döndüren fonksiyonlar***

Geri dönüş değeri pointer olan fonksiyonlar

#include <stdio.h>

**int**\* square(**int**\* ptr){ //pointer dönderen bir fonksiyon tanımladık

\*ptr=(\*ptr)\*(\*ptr); //tanım yaptıktan sonra \*ptr yaptığımızda değerden bahsediyoduk bu satırda kare aldık

**return** ptr; //ve sonucun adresini dönderdik

}

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** x=5;

**int**\* p=square(&x);//eşitliğin sağında call by reference ile x in adresini yolladik ve dönen adresi başka bir

//pointerda tuttuk.

printf("%d\n",\*p); //sonra da son pointerın değerini %d ile yazdırarak 25 sonucuna ulaştık.

**return** 0;

}

#include <stdio.h>

**int**\* sum(**int**\* s1,**int**\* s2){

**static** **int** result; //adresle donen fonksiyonlarda otomatik omurlu nesnelerle dönmememiz lazım

result=\*s1 + \*s2;

**return** &result;

}

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** a=10;

**int** b=20;

**int**\* ptr= sum(&a, &b);

printf("%d\n", \*ptr);

**return** 0;

}

***Array to pointer converter***

int a[5] diye bir dizimiz olsun.

Herhangi bir yerde a yazmamız &a[0] yazmamızla aynı şeydir ve dizinin ilk elemanının adresini belirtir.

bu sebeple int\* ptr = a gibi bir atama yapmamız mümkündür.

#include <stdio.h>

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** a[5]={1,2,3,4,5};

**int**\* p1=a;

**int**\* p2=&a[0];

printf("p1: %p\n",p1);//a[0] ın adresini göstermeli

printf("p2: %p\n",p2);//bu da a[0] ın adresini göstermeli

printf("\*p1: %d\n",\*p1);//a[0]ı yani 1 i göstermeli

printf("\*p2: %d\n",\*p2);//a[0]ı yani 1 i göstermeli

**return** 0;

}

**p1: 0x7ffeefbff4d0**

**p2: 0x7ffeefbff4d0**

**\*p1: 1**

**\*p2: 1**

***POINTER ARİTMETİĞİ***

* Bir pointer ile bir sayı toplanabilir
* Bir tam sayı ile bir pointerı toplanabilir
* Bir pointerdan bir tam sayıyı çıkarılabilir

(Bir tam sayıdan bir pointer çıkarılmaz)

* Bir pointerdan başka bir pointerı çıkarılabilir

#include <stdio.h>

**int** main() {

**int** a[]={2,4,6,8,10};

**int**\* ptr=a;

ptr=a+2; //dizinin 0.elemanının adresini tutuyodu şimdi 2 ileriki aelemanın adresini tutuyor.

printf("%d\n",\*ptr); // \*ptr yapınca da a[2] yi gösterdi

ptr-=1;//1 çıkaralım, bu sefer 4 ün adresini işaret ediyor.

printf("%d\n",\*ptr);

printf("ptr: %p\n",ptr); //4 ün adresini öğrenelim

ptr=ptr-1;//ilk elemanın adresine geri götürelim

printf("ptr: %p\n",ptr);

//şimdi 1 adım daha geriye gitsin bakalım hangi adresi ve hangi değeri gösterecek

ptr=ptr-1;

printf("ptr (ilk değerden bir adım geri): %p\n",ptr);

printf("\*ptr(ilk değerden bir adım gerideki değer): %d\n",\*ptr); //bnde 0 çıktı ama çöp değer de çıkabilirdi

//

**return** 0;

}

**6**

**4**

**ptr: 0x7ffeefbff4d4**

**ptr: 0x7ffeefbff4d0**

**ptr (ilk değerden bir adım geri): 0x7ffeefbff4cc**

**\*ptr(ilk değerden bir adım gerideki değer): 0**

**Program ended with exit code: 0**

#include <stdio.h>

**int** main() {

**int** a[]={2,4,6,8,10};

**int**\* p1=a;

**int**\* p2=a+2;

**int**\*p3=p1+p2; //GECERSİZ İSLEM

printf("%p\n",p3);//calismadi bile

printf("%d\n",\*p3);//bu da calismadi

**return** 0;

}

bunu sor

#include <stdio.h>

**int** main() {

**int** a[]={2,4,6,8,10};

**int**\* p1=a;

**int**\* p2=a+8;//a’nın yanındaki eleman kaçsa onu döndürüyor??(0x8, 0x4 gibi)

**int** \*p3=p2-p1;

printf("%p\n",p3);

printf("%d\n",\*p3);

**return** 0;

}

direkt dizi tanımlamadan pointer yoluyla bi diziyi tanimlayip elemanlarina ulaşabilirsin pointer yoluyla tanimladiginda zaten diziyi tanimlamis oluyosun ama işaret ederek yani.

**int** N,i;

**int** \*arr;

printf("N degerini girin:\n");

scanf(" %d",&N);

arr=(**int**\*)calloc(N,**sizeof**(**int**));

**for** (i=0; i<N; i++) {

printf("%d ", arr[i]);// dizi yoluyla yazdıralım basta bi gorelim icini

}

**for** (i=0; i<N; i++) { //dizi yoluyla scanfle alalım

printf("%d. eleman: \n", i+1);

scanf("%d", &arr[i]);

}

**for** (i=0; i<N; i++) {

printf("%d ", arr[i]);// dizi yoluyla yazdıralım yine

}

**for** (i=0; i<N; i++) { //pointer yoluyla alalım degerleri

printf("%d. eleman: \n", i+1);

scanf("%d", arr+i);

}

**for** (i=0; i<N; i++) { //yine pointer yoluyla yazdiralim

printf("%d ", \*(arr+i));

}

printf("\n");