**yapısal programlama 9.hafta dersi notlar (structures)**

**call by reference call by value**

#include <stdio.h>

**int** f(**int**,**int**\*,**int**\*);

**int** main(){

**int** a,b,c=6;

**int** x;

printf("%d %d %d %d",a,b,c,x); //g,g,6,g

x=f(c, &a, &b);

printf("\n");

printf("%d %d %d %d",a,b,c,x);//12,10,6,14

}

**int** f(**int** a,**int** \*pa,**int** \*pb){

**int** x=5;

**int** y=7;

a+=8;

\*pa=x+7;

\*pb=y+3;

**return** a;

}

başına const getirdiğin bir şeyi artık değiştiremezsin:

mesela const int m=5; dedik sonra m=6; yaptığımızda syntax hatası

const neyin önüne gelmişse o sabit oluyor değiştirilemez oluyor.

**int** x=20;

**int** y=30;

**int** \* **const** ptr=&x;

ptr=&y;

//hata alırız çünkü ptr sadece x in adresini tutsun dedik ama sonra gidip y nin adresine atadık.

**int** x=20;

**int** y=30;

**int** **const** \*ptr=&x;

ptr=&y;//ptr a bir şey atayabiliriz çünkü \*ptr sabit

\*ptr=x; //yapamayız

\*ptr=25; //böyle de yapamayız

int const \* ptr yaparsan ptr ı sabitler

const int \*ptr yaparsan \*ptr ı sabitler yani içeriği

const int const \* ptr yaparsan da hem değeri hem adresi sabitler

not:

**int** \*f(){

**int** x;

**static** **int** a;

**int** \*p=(**int**\*)malloc.....

**return** &x //olur

&a //olur

p //olur

//ama static oldugu icin a da ve mallocla tanımlandıgı icin p'de sorun cıkmazken x'te cıkabilir cunku

//omru sınırlı, digerleri program kapanana kadar calisir (malloclu olan free edilmedigi surece)

//yani main icinde int\*y=f() yaptıgımızda a ya ve p ye program kapanana kadar erisecekken x e ise erisememe ihtimali vardır

//cunku fonksiyon tamamlandıgında x degiskeni olecektir

//SONUC OLARAK bir pointer fonksiyon tanımladıgında geriye neyi dondurdugune dikkat et.

}

örnek:mesela bunda uyarı geçmedi çünkü static bir şeye eriştik ve program kapanana kadar da erişir.

#include <stdio.h>

**int** \*f(**void**);

**int** main(){

**int** \*g=f();

printf("%d",\*g);

}

**int** \*f(){

**int** x=20;

**static** **int** y=30;

**return** &y;

}

bunda ise uyarı verdi local bir değişkene ulaşmaya calisiyorsun seklinde (ömrü kısıtlı olduğundan hep aynı değere ulaşamayabiliriz.)

#include <stdio.h>

**int** \*f(**void**);

**int** main(){

**int** \*g=f();

printf("%d",\*g);

}

**int** \*f(){

**int** x=20;

**static** **int** y=30;

**return** &x;

}

psödo kod şeklinde (kullanımları göstermek için. hepsi doğru)

#include <stdio.h>

#define MAX 10

**int** main(){

**int** dizi[MAX];

dizi\_yaz(dizi);

dizi\_yaz1(dizi,MAX);

dizi\_yaz2(&dizi[0],MAX);

dizi\_yaz2(dizi,MAX);

dizi\_yaz2(&dizi,MAX);

**return** 0;

}

**void** dizi\_yaz(**int** dizi[MAX]){

dizi[i]

}

**void** dizi\_yaz1(**int** dizi[],MAX){

dizi[i]

}

**void** dizi\_yaz2(**int** \*dizi,MAX){

dizi[i]

}

* #define SIZE\_OF\_ARRAY(x) (sizeof(x)/ sizeof((x)[0]) ) (dizinin size’ını herhangi bir elemanın boyutuna bölersen kaç adet elemanı olduğunu anlarsın.)
* aşağıdaki gib bir fonksiyon tipi yok!!

int[ ]sort (){

}

ÖRNEK:

#include <stdlib.h>

#include <stdio.h>

**int**\* concatArrays(**int**[],**int**, **int**[],**int**);

**int** main(){

**int** array1[]={1,2,3,4,5};

**int** array2[]={6,7,8,9,10,11,12};

**int** i;

**int** s1=(**sizeof**(array1)/**sizeof**(**int**));

printf("%d\n",s1);

**int** s2=(**sizeof**(array2)/**sizeof**(**int**));

printf("%d\n",s2);

**int** \*ptr= concatArrays(array1, s1, array2, s2);

**for** (i=0; i<s1+s2; i++) {

printf("%d ",ptr[i]);

}

**return** 0;

}

**int** \* concatArrays(**int** arr1[],**int** size1, **int** arr2[], **int** size2){

**int** \*merged=(**int**\*)malloc((size1+size2)\***sizeof**(**int**));

**int** i;

**for** (i=0; i<size1; i++) {

merged[i]=arr1[i];

}

**for** ( i=size1; i<size1+size2; i++) {

merged[i]=arr2[i-size1];

}

**return** merged;

}

**5**

**7**

**1 2 3 4 5 6 7 8 9 10 11 12 Program ended with exit code: 0**

1.saat 7. dk (3 saat 54 dklık, sonunda sınav çözümleri var)

argc argv bi seyler yapildi anlamadım önemli mi?

STRUCTURES AND UNIONS

struct Employee{

char name[20], surname[20];

int ID;

float salary;

};

bu deklarasyondan sonra

* struct Employee e; // int a gibi, struct employee burda bir tür artık
* struct Employee \*e; //dinamik bi dizi (pointer yoluyla)

e=malloc(N\*sizeof(struct Employee));

* struct Employee employees[1000]; //statik bi dizi int a[100] der gibi

sürekli struct dememek için typedef kullanabiliriz

mesela typedef int tamsayi yapsan artik int yerine tamsayi yazabilirsin.

typedef struct {

char name[20], surname[20];

int ID;

float salary;

}EMPLOYEE;

artık struct yerine EMPLOYEE yazabilirsin.

EMPLOYEE e;

EMPLOYEE emps[1000]; yapabilirim

örnek

#include <stdio.h>

// polinom= a1\*x^(n)+a2\*x^(n-1)+a3\*x^(n-2)+......+an\*x^(0)

**typedef** **struct** polinom{

**int** katsayi;

**int** derece;

}POLINOM;// boyle yaparsak hem struct diyerek hem de demeksizin tanımlama yapabiliriz.

**int** main(){

**struct** polinom p1;

POLINOM p2;

}

POLINOM p1;

POLINOM \*p2;

POLINOM \*\*p3;

POLINOM p4[10];

POLINOM \*p5[10];

POLINOM p6[4][7];//bu tanımlamaların hepsi mümkün

bu polinomun elemanlarına nokta operatörüyle erişebiliriz.

//veri girişi ve verilere erişim

#include <stdio.h>

// polinom= a1\*x^(n)+a2\*x^(n-1)+a3\*x^(n-2)+......+an\*x^(0)

**typedef** **struct** polinom{

**int** derece;

**double** katsayi;

}POLINOM;// boyle yaparsak hem struct diyerek hem de demeksizin tanımlama yapabiliriz.

**int** main(){

POLINOM p1;

POLINOM \*p2;

POLINOM \*\*p3;

POLINOM p4[10];

POLINOM \*p5[10];

POLINOM p6[4][7];

p1.derece=4;

p1.katsayi=6;

p2=&p1;

printf("Derece: %d, Katsayi:%2.2lf\n",p1.derece,p1.katsayi);

printf("Struct adres: %p, derece adres:%p, katsayi adres: %p ",&p1,&p1.derece,&p1.katsayi);

printf("\nDerece giriniz: ");

scanf(" %d",&p1.derece);

printf("Katsayi giriniz: ");

scanf(" %lf",&p1.katsayi);

printf("Derece: %d Katsayi:%2.2lf\n",p1.derece,p1.katsayi);

**return** 0;

}

dot ve arrow operatörleri

dot:

yapı\_turunden\_nesne.yapı\_turunden\_eleman

arrow:

yapı\_turunde\_pointer\_nesne->yapı\_turunden\_eleman

//yukardaki orneklerin genişletilmiş hali

#include <stdio.h>

// polinom= a1\*x^(n)+a2\*x^(n-1)+a3\*x^(n-2)+......+an\*x^(0)

**typedef** **struct** polinom{

**int** derece;

**double** katsayi;

}POLINOM;// boyle yaparsak hem struct diyerek hem de demeksizin tanımlama yapabiliriz.

**int** main(){

POLINOM p1;

POLINOM \*p2;

POLINOM \*\*p3;

POLINOM p4[10];

POLINOM \*p5[10];

POLINOM p6[4][7];

p1.derece=17;

p1.katsayi=6;

p2=&p1;

p3=&p2;

//dot operatoruyle erisim

printf("Derece: %d, Katsayi:%2.2lf\n",p1.derece,p1.katsayi);

printf("Struct adres: %p, derece adres:%p, katsayi adres: %p \n",&p1,&p1.derece,&p1.katsayi);

printf("Struct adres: %p, derece adres:%p, katsayi adres: %p \n",&p1,&p2->derece,&p2->katsayi);

printf("Struct adres: %p, derece:%d, katsayi: %2.2lf \n",&p1,(\*p2).derece,(\*p2).katsayi);

//decimal bi seyi %p ile yazdirmaya calisirsan hexadecimal yazıyor burda ornegi yok da soylemek istedim

printf("Struct adres: %p, derece adres:%p, katsayi adres: %p \n",&p1,&(\*p2).derece,&(\*p2).katsayi);

printf("\nDerece giriniz: ");

scanf(" %d",&p1.derece);

printf("Katsayi giriniz: ");

scanf(" %lf",&p1.katsayi);

printf("Derece: %d Katsayi:%2.2lf\n",p1.derece,p1.katsayi);

(\*p2).derece=60;

p2->katsayi=44;

printf("Derece: %d Katsayi:%2.2lf\n",p1.derece,p1.katsayi);

//arrow operatoruyle erisim

printf("\nDerece giriniz: ");

scanf(" %d",&p2->derece); //& koymamız gerekiyor çünkü p2 bir adres ama p2->derece diyince bir deger oldu ve scanfte deger degil adres koymalısın virgulden sonra

printf("Katsayi giriniz: ");

scanf(" %lf",&p2->katsayi);

printf("Derece: %d Katsayi:%2.2lf\n",p2->derece,p2->katsayi);

**return** 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// polinom= a1\*x^(n)+a2\*x^(n-1)+a3\*x^(n-2)+......+an\*x^(0)

**typedef** **struct** terim{

**double** katsayi;

**int** derece;

}TERIM;

**typedef** **struct** polinom{

TERIM \*terimler;

**int** terim\_sayisi;

}POLINOM;// boyle yaparsak hem struct diyerek hem de demeksizin tanımlama yapabiliriz.

**int** main(){

srand(time(**NULL**));

POLINOM p1;

POLINOM \*p2;

p2=&p1;

**int** N;

printf("Terim sayisini giriniz: \n");

scanf(" %d",&N);

p1.terim\_sayisi=N;

p1.terimler=(TERIM\*)malloc(N\***sizeof**(TERIM));

**int** i;

//otomatik atansın istersen

**for** (i=0; i<p1.terim\_sayisi; i++) {

p1.terimler[i].derece=i;

p1.terimler[i].katsayi=rand()%4+1;

}

**for** (i=0; i<p1.terim\_sayisi-1; i++) { //print ederken sonda fazladan bir + gormemek icin

printf("%2.2lf X^%d +", p1.terimler[i].katsayi,p1.terimler[i].derece);

}

//kendin koymak istersen

printf("%2.2lf X ^ %d\n", p1.terimler[i].katsayi,p1.terimler[i].derece);

**for** (i=0; i<p1.terim\_sayisi; i++) {

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

scanf(" %d", &p1.terimler[i].derece);

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

scanf(" %lf", &p1.terimler[i].katsayi);

}

**for** (i=0; i<p1.terim\_sayisi-1; i++) { //print ederken sonda fazladan bir + gormemek icin

printf("%2.2lf X^%d +", p1.terimler[i].katsayi,p1.terimler[i].derece);

}

printf("%2.2lf X ^ %d\n", p1.terimler[i].katsayi,p1.terimler[i].derece);

//pointer yoluyla yazdırmak istersek

**for** (i=0; i<p1.terim\_sayisi-1; i++) { //print ederken sonda fazladan bir + gormemek icin

printf("%2.2lf X^%d +", p2->terimler[i].katsayi,(\*p2).terimler[i].derece);

}

printf("%2.2lf X ^ %d\n", p2->terimler[i].katsayi,(\*p2).terimler[i].derece);

//pointer kullanırken scanfle almak istersek de böyle

//yine & kullandık cünkü p2->terimler[i].derece ya da katsayi yaptigimizda yine bir degere ulasmis oluyoruz

**for** (i=0; i<p1.terim\_sayisi; i++) {

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

scanf(" %d", &p2->terimler[i].derece);

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

scanf(" %lf", &p2->terimler[i].katsayi);

}

//sonra yine yazdirma

**for** (i=0; i<p1.terim\_sayisi-1; i++) { //print ederken sonda fazladan bir + gormemek icin

printf("%2.2lf X^%d +", p2->terimler[i].katsayi,(\*p2).terimler[i].derece);

}

printf("%2.2lf X ^ %d\n", p2->terimler[i].katsayi,(\*p2).terimler[i].derece);

**return** 0;

}

//buna tekrar bak anlamadım.

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// polinom= a1\*x^(n)+a2\*x^(n-1)+a3\*x^(n-2)+......+an\*x^(0)

**typedef** **struct** terim{

**double** katsayi;

**int** derece;

}TERIM;

**typedef** **struct** polinom{

TERIM \*terimler;

**int** terim\_sayisi;

}POLINOM;// boyle yaparsak hem struct diyerek hem de demeksizin tanımlama yapabiliriz.

**int** main(){

srand(time(**NULL**));

POLINOM p1;

POLINOM \*p2;

p2=&p1;

TERIM t1;

t1.derece=5;

t1.katsayi=3.6;

p1.terimler=&t1;

printf("%d %2.2lf\n", p1.terimler->derece,p1.terimler->katsayi);

printf("%d %2.2lf\n", p2->terimler->derece,p2->terimler->katsayi);

printf("%d %2.2lf\n", (\*p2).terimler->derece,(\*p2).terimler->katsayi);

**return** 0;

}

// kendi notum

#include <stdio.h>

#include <stdlib.h>

// polinom= a1\*x^(n)+a2\*x^(n-1)+a3\*x^(n-2)+......+an\*x^(0)

**typedef** **struct** polinom{

**int** derece;

**double** katsayi;

}POLINOM;// boyle yaparsak hem struct diyerek hem de demeksizin tanımlama yapabiliriz.

**int** main(){

**int** N;

**int** i;

printf("Polinomunuz kac elemanli olacak?:\n");

scanf(" %d", &N);

POLINOM \*pol=(POLINOM \*)malloc(N\***sizeof**(POLINOM));

//dot operatoruyle deger alimi

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

printf("Pol[%d] nin katsayisi: \n", i+1);

scanf(" %lf", &(pol[i].katsayi));

printf("Pol[%d] nin derecesi: \n", i+1);

scanf(" %d", &(pol[i].derece));

}

//N-1 e kadar yazdirim (+ bozmasın diye) (dot operatoru ile)

**for** (i=0; i<N-1; i++) {

printf("%2.2lf X ^", pol[i].katsayi);

printf(" %d + ", pol[i].derece);

}

//son eleman yazdiriliyor (dot operatoru ile)

printf("%2.2lf X ^", pol[N-1].katsayi);

printf("%d", pol[N-1].derece);

//arrow ile deger alimi

printf("\n");

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

printf("Pol[%d] nin katsayisi: \n", i+1);

scanf("%lf", &((pol+i)->katsayi));

printf("Pol[%d] nin derecesi: \n", i+1);

scanf(" %d", &((pol+i)->derece));

}

//+ isareti bozmasin diye son elemana kadar olan kısım yazdırılıyor (arrow operetru ile)

**for** (i=0; i<N-1; i++) {

printf("%2.2lf X ^", (pol+i)->katsayi);

printf(" %d +", (pol+i)->derece);

}

//son eleman yazdırılıyor (arrow operatoru ile)

printf("%2.2lf X ^", (pol+N-1)->katsayi);

printf("%d", (pol+N-1)->derece);

**return** 0;

}