**Veri Yapıları Ödevi :**

Bu verileri veri yapıları türlerini açıklayan C dilinde çözen kodlar yazınız.

1. Çift Yönlü Bağlı Listeler Ekleme, Silme, Arama, Düze Listeme, Ters Listeleme

#include <stdio.h>

#include <stdlib.h>

//sadece menü için global değişken

int choise;

//düğümlerimizi tutacağımız struct yapısı

struct node{

 int data;

 struct node \*next;

};

//Başlangıç düğümü(start) ve geçici değişken (temp) oluşturuyoruz

struct node\* start = NULL;

struct node\* temp;

//düğüm oluşturan fonksiyon, geriye düğüm (node) return ediyor

struct node\* createNode()

{

 struct node\* newNode = (struct node\*)malloc(sizeof(struct node));

 return newNode;

}

//Bağlı liste sonuna eleman ekleyen fonksiyon

void addLast(int x)

{

 struct node\* element = createNode();

 element->data = x;

 element->next = NULL;

 //Eğer bağlı liste zaten boşsa ilk düğümü ekliyoruz

 if(start == NULL)

 {

 start = element;

 }

 else

 {

 //aşağıdaki kodlar traverse yapıp düğümün sonunu buluyor ve yeni düğümü ekliyor

 temp = start;

 while(temp->next != NULL)

 {

 temp = temp->next;

 }

 temp->next = element;

 }

}

//Bağlı listenin başına eleman ekleyen fonksiyon

void addFirst(int y)

{

 struct node\* element = createNode();

 element->data = y;

 temp = start;

 element->next = temp;

 start = element;

}

//Spesifik bir pozisyona eleman ekleyen fonksiyon, belirttiğiniz pozisyonun bir öncesinde bulunan düğümü bulup onun önüne düğümü ekliyor

void addPos(int z, int index)

{

 int counter = 0;

 struct node\* anywhere = createNode();

 anywhere->data = z;

 temp = start;

 if(index == 0)

 {

 addFirst(z);

 }

 else

 {

 while(temp != NULL)

 {

 if(counter+1 == index)

 break;

 temp = temp->next;

 counter++;

 }

 struct node\* temp1;

 temp1 = temp->next;

 temp->next = anywhere;

 anywhere->next = temp1;

 }

}

//Bağlı liste sonundan eleman silen fonksiyon.

void deleteLast()

{

 if(start == NULL)

 {

 printf("\n Your List is EMPTY");

 }

 else

 {

 temp = start;

 if(start->next == NULL)

 {

 free(start);

 start = NULL;

 }

 else

 {

 while(temp->next->next != NULL)

 {

 temp = temp->next;

 }

 struct node\* cukubik = temp->next;

 free(cukubik);

 temp->next = NULL;

 }

 }

}

//Bağlı listenin başından eleman silen fonksiyon

void deleteFirst()

{

 if(start != NULL)

 {

 if(start->next != NULL)

 {

 struct node\* temp2 = start;

 start = start->next;

 free(temp2);

 }

 else

 {

 free(start);

 start = NULL;

 }

 }

 else

 {

 printf("\nThere is no item to be deleted, Please add item...");

 }

}

//Belirttiğiniz sayıyı silen fonksiyon

void deleteSpes(int t)

{

 if(start->data == t)

 {

 deleteFirst();

 }

 else

 {

 temp = start;

 while(temp->next->data != t)

 {

 temp = temp->next;

 }

 struct node\*temp1 = temp->next->next;

 struct node\* temp2 = temp;

 free(temp->next);

 temp2->next = temp1;

 }

}

//Bağlı listeyi ters çeviren fonksiyon

void reverseLinked()

{

 struct node\* prevNode = NULL;

 struct node\* nextNode;

 temp = start;

 while(temp != NULL)

 {

 nextNode = temp->next;

 temp->next = prevNode;

 prevNode = temp;

 temp = nextNode;

 }

 start = prevNode;

}

//Bağlı listeyi ekrana basan fonksiyon

void printLinked()

{

 if(start == NULL)

 {

 printf("\n Your List is EMPTY");

 }

 else

 {

 temp = start;

 printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

 while(temp->next != NULL)

 {

 printf("%d ", temp->data);

 temp = temp->next;

 }

 printf("%d ", temp->data);

 printf("\n");

 }

}

void menu()

{

 while(1 == 1)

 {

 printf("\n1- Bagli listenin sonuna eleman ekle ");

 printf("\n2- Bagli listenin basina eleman ekle ");

 printf("\n3- Herhangi bir pozisyona eleman ekle ");

 printf("\n4- Bagli Liste sonundaki elemani sil ");

 printf("\n5- Bagli Liste basindaki elemani sil ");

 printf("\n6- Herhangi bir sayiyi sil ");

 printf("\n7- Bagli listeyi ters cevir");

 printf("\n Secimini yap");

 scanf("%d", &choise);

 selection(choise);

 }

}

//Menüden yapılan seçimi handle eden fonksiyon

void selection(int choise)

{

 int item, i, index;

 switch(choise)

 {

 case 1:

 printf("\n Kac Tane Sayi Ekleyeceksiniz? ... ");

 scanf("%d", &item);

 for(i = 0; i < item; i++)

 {

 printf("\n Lutfen %d. sayiyi ekleyin... ", i+1);

 scanf("%d", &choise);

 addLast(choise);

 printLinked();

 }

 break;

 case 2:

 printf("En Ba$a eklemen istediginiz sayiyi girin? ... ");

 scanf("%d", &item);

 addFirst(item);

 printLinked();

 break;

 case 3:

 printf("Hangi sayiyi eklemek istiyorsunuz? ...");

 scanf("%d", &item);

 printf("Hangi indexe yerlestirmek istiyorsunuz? (ilk sayi 0. indexte) ...");

 scanf("%d", &index);

 addPos(item, index);

 printLinked();

 break;

 case 4:

 deleteLast();

 printLinked();

 break;

 case 5:

 deleteFirst();

 printLinked();

 break;

 case 6:

 printf("\n Hangi sayiyi silmek istiyorsunuz? ...");

 scanf("%d", &item);

 deleteSpes(item);

 printLinked();

 break;

 case 7:

 reverseLinked();

 printLinked();

 }

}

//Main fonksiyonumuz

int main()

{

 menu();

}

1. AVL Tree (Dengeli Ağaç ) Ekleme, Silme, Arama, Düze Listeme

#include <stdio.h>

#include <stdlib.h>

typedef struct Agac{

 int sayi,seviye;

 struct Agac \*sol;

 struct Agac \*sag;

}agac;

agac \*kok=NULL,\*baba,\*a;

int yon;

int sonseviye=-1;

int derinlik;

agac \*Bilgi\_Al(){

 agac \*p=(agac \*)malloc(sizeof(agac));

 printf("Bir sayi giriniz:\n");

 scanf("%d",&(p->sayi));

 p->sol=NULL;

 p->sag=NULL;

 return p;

}

int kontrol(agac \*p,agac \*gecici){

 printf("Kontrol ediliyor...\n");

 if((p->sayi)<(gecici->sayi)){

 if((gecici->sol)==NULL){

 gecici->sol=p;

 printf("Veri eklendi...\n");

 return 1;

 }else{

 gecici=gecici->sol;

 kontrol(p,gecici);

 }

 }else if((p->sayi)>(gecici->sayi)){

 if((gecici->sag)==NULL){

 gecici->sag=p;

 printf("Veri eklendi...\n");

 return 1;

 }else{

 gecici=gecici->sag;

 kontrol(p,gecici);

 }

 }else{

 printf("Bu sayi zaten kayitli!!!\n");

 return 0;

 }

 return 0;

}

void Ekle(agac \*p){

 printf("Ekleme islemi baslatiliyor...\n");

 if(p){

 if(kok==NULL){ //Kayitli eleman yoksa

 kok=p;

 printf("Kok olusturuldu...\n");

 }else{ //Kayitli eleman varsa

 kontrol(p,kok);

 }

 }

}

agac \*Arama(int ara,agac \*gecici){

 if(gecici){

 if((gecici->sayi)>ara){

 baba=gecici;

 yon=-1;

 gecici=gecici->sol;

 a=Arama(ara,gecici);

 return a;

 }else if((gecici->sayi)<ara){

 baba=gecici;

 yon=1;

 gecici=gecici->sag;

 a=Arama(ara,gecici);

 return a;

 }else if((gecici->sayi)==ara){

 printf("Aradiginiz kayit bulundu...\n");

 printf("RAM adresi: %p\n",gecici);

 printf("Babanin RAM adresi: %p\n",baba);

 return gecici;

 }

 }else{

 printf("Aradiginiz kayit bulunamadi...\n");

 return NULL;

 }

return a;

}

void Sil(agac \*ptr){

 printf("1\n");

 if(ptr){

 if(kok==ptr){

 printf("a\n");

 if(ptr->sag){

 printf("1\n");

 kok=ptr->sag;

 Ekle(ptr->sol);

 }else if(ptr->sol){

 printf("2\n");

 kok=ptr->sol;

 Ekle(ptr->sag);

 }else{

 printf("3\n");

 kok=NULL;

 }

 }else{

 printf("b\n");

 agac \*SOL=ptr->sol,\*SAG=ptr->sag;

 if(yon==-1){

 printf("1\n");

 baba->sol=NULL;

 if(SAG){

 Ekle(SAG);

 printf("Sag Eklendi...\n");

 }

 if(SOL){

 Ekle(SOL);

 printf("Sol Eklendi...\n");

 }

 yon=0;

 }else if(yon==1){

 printf("2\n");

 baba->sag=NULL;

 if(SAG){

 Ekle(SAG);

 printf("Sag Eklendi...\n");

 }

 if(SOL){

 Ekle(SOL);

 printf("Sol Eklendi...\n");

 }

 yon=0;

 }

 }

 baba=kok;

 free(ptr);

 }else{

 printf("Silinecek veri yok...\n");

 }

}

void preorder(agac \*gecici){

 if(gecici!=NULL){

 printf("%d\n",gecici->sayi);

 preorder(gecici->sol);

 preorder(gecici->sag);

 }

}

void inorder(agac \*gecici){

 if(gecici!=NULL){

 inorder(gecici->sol);

 printf("%d\n",gecici->sayi);

 inorder(gecici->sag);

 }

}

void postorder(agac \*gecici){

 if(gecici!=NULL){

 postorder(gecici->sol);

 postorder(gecici->sag);

 printf("%d\n",gecici->sayi);

 }

}

int Derinlik(agac \*gecici){

 int sol,sag;

 if(gecici){

 sol=Derinlik(gecici->sol);

 sag=Derinlik(gecici->sag);

 if(sag>sol){

 return sag+1;

 }else{

 return sol+1;

 }

 }else return 0;

}

void Seviye\_Ver(agac \*gecici){

 sonseviye++;

 gecici->seviye=sonseviye;

 if(gecici->sol){

 Seviye\_Ver(gecici->sol);

 }

 if(gecici->sag){

 Seviye\_Ver(gecici->sag);

 }

 sonseviye--;

}

void Seviye\_Listele(int i,agac \*gecici){

 if(gecici!=NULL){

 Seviye\_Listele(i,gecici->sol);

 if(gecici->seviye==i){

 printf("%d, ",gecici->sayi);

 }

 Seviye\_Listele(i,gecici->sag);

 }

}

int AVL\_Tespit(agac \*gecici){

 int sol,sag,solalt,sagalt;

 agac \*gecalt;

 agac \*j;

 if(gecici){

 sol=AVL\_Tespit(gecici->sol);

 sag=AVL\_Tespit(gecici->sag);

 printf("%d noktasının sagi ve solunun yuksekligi arasındaki fark: %d\n",gecici->sayi,sol-sag);

 if(((sol-sag)>1)||((sol-sag)<-1)){

 printf("Bu bir AVL agaci degil...\n");

 //Bozuklugun tipini belirle

 if((sol-sag)>0){//sol-sag>0 ise sol daha yuksek

 //solun solu mi buyuk yosa sagi mu

 gecalt=gecici->sol;

 solalt=AVL\_Tespit(gecalt->sol);

 sagalt=AVL\_Tespit(gecalt->sag);

 if(solalt>sagalt){

 //solun solu proseduru

 //SolSol();

 printf("Solun Solu\n");

 if(gecici==kok){

 kok=gecalt;

 gecici->sol=gecalt->sag;

 gecalt->sag=gecici;

 //Ekle(gecici);

 }else{

 Arama(gecici->sayi,kok);

 if(yon==-1){

 baba->sol=gecalt;

 gecici->sol=gecalt->sag;

 gecalt->sag=gecici;

 }else if(yon==1){

 baba->sag=gecalt;

 gecici->sol=gecalt->sag;

 gecalt->sag=gecici;

 }

 }

 }else{

 //solun sagi proseduru

 //SolSag(gecalt,gecici);

 printf("Solun Sagi\n");

 j=gecalt->sag;

 gecici->sol=j;

 gecalt->sag=j->sol;

 j->sol=gecalt;

 gecalt=j;

 if(gecici==kok){

 kok=gecalt;

 gecici->sol=gecalt->sag;

 gecalt->sag=gecici;

 }else{

 Arama(gecici->sayi,kok);

 if(yon==-1){

 baba->sol=gecalt;

 gecici->sol=gecalt->sag;

 gecalt->sag=gecici;

 }else if(yon==1){

 baba->sag=gecalt;

 gecici->sol=gecalt->sag;

 gecalt->sag=gecici;

 }

 }

 //return sag;

 }

 sol--;

 }else if((sol-sag)<0){//sol-sag<0 ise sag daha yuksek

 //sagin sagi mi buyuk yosa solu mu

 gecalt=gecici->sag;

 solalt=AVL\_Tespit(gecalt->sol);

 sagalt=AVL\_Tespit(gecalt->sag);

 if(sagalt>solalt){

 //sagin sagi proseduru

 printf("Sagin Sagi\n");

 if(gecici==kok){

 kok=gecalt;

 gecici->sag=gecalt->sol;

 gecalt->sol=gecici;

 //Ekle(gecici);

 }else{

 Arama(gecici->sayi,kok);

 if(yon==-1){

 baba->sol=gecalt;

 gecici->sag=gecalt->sol;

 gecalt->sol=gecici;

 }else if(yon==1){

 baba->sag=gecalt;

 gecici->sag=gecalt->sol;

 gecalt->sol=gecici;

 }

 }

 }else{

 //sagin solu proseduru

 printf("Sagin Solu\n");

 j=gecalt->sol;

 gecici->sag=j;

 gecalt->sol=j->sag;

 j->sag=gecalt;

 gecalt=j;

 if(gecici==kok){

 kok=gecalt;

 gecici->sag=gecalt->sol;

 gecalt->sol=gecici;

 }else{

 Arama(gecici->sayi,kok);

 if(yon==-1){

 baba->sol=gecalt;

 gecici->sag=gecalt->sol;

 gecalt->sol=gecici;

 }else if(yon==1){

 baba->sag=gecalt;

 gecici->sag=gecalt->sol;

 gecalt->sol=gecici;

 }

 }

 //return sol;

 }

 sag--;

 }

 }

 if(sag>sol){

 return sag+1;

 }else{

 return sol+1;

 }

 }else return 0;

}

int main(){

 char secim;

 int ara,sil;

 agac \*p;

 while(1){

 printf("Ekle(e) Arama(a) Sil(s) Preorder(r) Inorder(i) Postorder(o) Derinlik(d) Seviyeler(l) AVL(v) Cikis(c)\n");

 do{

 secim=getchar();

 }while(secim=='\n');

 switch(secim){

 case 'e':

 p=Bilgi\_Al();

 Ekle(p);

 break;

 case 'a':

 printf("Aranacak sayiyi giriniz:\n");

 scanf("%d",&ara);

 Arama(ara,kok);

 break;

 case 's':

 printf("Silinecek sayiyi giriniz:\n");

 scanf("%d",&sil);

 p=Arama(sil,kok);

 Sil(p);

 break;

 case 'r':

 preorder(kok);

 break;

 case 'i':

 inorder(kok);

 break;

 case 'o':

 postorder(kok);

 break;

 case 'd':

 printf("Derinlik: %d\n",Derinlik(kok)-1);

 break;

 case 'l':

 derinlik=Derinlik(kok);

 Seviye\_Ver(kok);

 int i;

 for(i=0;i<derinlik;i++){

 printf("Seviye %d:",i);

 Seviye\_Listele(i,kok);

 printf("\n");

 }

 break;

 case 'v':

 AVL\_Tespit(kok);

 break;

 case 'c':

 exit(0);

 break;

 }

 }

 return 0;

}

1. Yığıt (Stack) Ekleme, Silme, Arama, Düze Listeme

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

struct stack{

 int top;

 int deger[10];

};

struct stack s;

void push(struct stack \*ps, int x){

 if(ps->top==9)

 printf("stack dolu");

 else

 ps->deger[(ps->top)++]=x;

}

void pop(struct stack \*ps){

 if(ps->top==0)

 printf("stack bos");

 else

 ps->top--;

}

int main()

{

 int sayi=0;

 while(sayi!=4){

 printf("Lutfen bir secim yapin: \n 1. Ekleme\n 2. Silme \n 3. Listeleme\n 4. Cikis\n");

 scanf("%d",&sayi);

 switch(sayi){

 case 1:

 printf("Eklemek istediginiz sayiyi girin:");

 int ekle;

 scanf("%d",&ekle);

 push(&s,ekle);

 break;

 case 2:

 pop(&s);

 break;

 case 3:

 if(s.top==0)

 printf("Liste bos\n");

 else

 printf("\nListe:");

 for(int a=0;a<s.top;a++)

 printf("%d ",s.deger[a]);

 printf("\n");

 break;

 }

 }

 getchar();

 return 0;

}

1. Kuyruk (Queue) Ekleme, Silme, Arama, Düze Listeme

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

//Düğümde ad soyad ve yaş değerlerini tutuyoruz.

struct node{

 char name[20];

 char secondName[20];

 int age;

 struct node \*next;

};

//Head: Kuyruğun ilk elemanı

//Tail: kuyruğun son elemanı

//temp: fonksiyonlarda kullanabilmemiz için geçici değişken

struct node\* head = NULL;

struct node\* tail = NULL;

struct node\* temp;

//İsim, soyisim ve yaş için global değişkenler

char firstName[20], secondName[20];

int age;

//Bu fonksiyon düğüm oluşturur ve return eder

struct node\* createPerson(char name[], char secondName[], int age)

{

 struct node\* newPerson = (struct node\*)malloc(sizeof(struct node));

 strcpy(newPerson->name, name);

 strcpy(newPerson->secondName, secondName);

 newPerson->age = age;

 newPerson->next = NULL;

 return newPerson;

};

//Kuyruğa eleman ekleme (enqueue)

void enQueuePerson(char name[], char secondName[], int age)

{

 struct node\* person = createPerson(name, secondName, age);

 //Eğer ilk eleman ve son eleman boş ise, yeni gelen eleman ilk elemandır

 if(head == NULL && tail == NULL)

 {

 head = person;

 tail = person;

 }

 else

 {

 //Sonuncunun arkasına eleman eklenir

 tail->next = person;

 //Sonuncu eleman yeni eklenen eleman olacak şekilde güncellenir.

 tail = person;

 }

}

//Kuyruktan eleman çıkaran fonksiyon

void deQueue()

{

 temp = head;

 //Eğer kuyruğun ilk elemanı yoksa kuyruk zaten boştur.

 if(head == NULL)

 {

 printf("\nQueue is empty, pls Enqueue");

 return;

 }

 //Eğer kuyruğun en arkasındaki kişi aynı zamanda en önündeki kişi ise kuyrukta 1 kişi var demektir

 //Bu yüzden o kişi de kuyruktan çıktığında kuyruğun hiçbir elemanı kalmaz.

 if(head == tail)

 {

 head = NULL;

 tail = NULL;

 return;

 }

 //Eğer yukarıdaki iki ihtimalde sağlanmazsa aşağıdaki kısım çalışır.

 //Mantık kuyruğun ikinci sırasındaki kişi artık birinci (head) sırasına gelmiş olur

 head = temp->next;

 //Free komutu ile düğümü yok ediyoruz

 free(temp);

}

//Sırada kimin olduğunu gösteren fonksiyon

struct node\* whoNext()

{

 if(head == NULL)

 {

 printf("\nThere is no item in queue...");

 return 0;

 }

 return head;

}

//Kuyruğu yazdıran fonksiyon

void printQueue()

{

 int i = 1;

 if(head == NULL)

 {

 return;

 }

 temp = head;

 while(temp->next != NULL)

 {

 printf("\n%d. Position => %s %s %d ", i, temp->name, temp->secondName, temp->age);

 temp = temp->next;

 i++;

 }

 printf("\n%d. Position => %s %s %d ", i, temp->name, temp->secondName, temp->age);

}

//menu fonksiyon

void menu()

{

 int choise;

 while( 1 == 1 )

 {

 printf("\n 1- Enqueue ... ");

 printf("\n 2- Dequeue ... ");

 printf("\n 3- Who's next? ");

 printf("\nmake your choise ");

 scanf("%d", &choise);

 selection(choise);

 }

}

//Menu secimi

void selection(int chosen)

{

 switch(chosen)

 {

 case 1:

 printf("\n Enter First Name ... ");

 scanf("%s", &firstName);

 printf("\n Enter Second Name ... ");

 scanf("%s", &secondName);

 printf("\n Enter Age ... ");

 scanf("%d", &age);

 enQueuePerson(firstName, secondName, age);

 printQueue();

 break;

 case 2:

 deQueue();

 printQueue();

 break;

 case 3:

 temp = whoNext();

 printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

 if(temp != NULL)

 {

 printf("%s %s %d", temp->name, temp->secondName, temp->age);

 }

 break;

 }

}

//Main fonksiyonu, yalnızca menu fonksiyonunu çağırıyoruz.

int main()

{

 menu();

 return 0;

}

Ödevde olması istenenler
1 Kapak Sayfası
2 Giriş
3 Yukarıda kodlamasını yapacağınız Türlerin genel açıklaması
4 Herbir tür için çalışan C kodları ve çalıştığına dair ekran çıktıları içerecek

Dosyadı : Veri\_Yapılari\_Odevi\_Bahattin\_Parlak\_9879878979.pdf şeklinde olmalıdır.

Önemli Not: Kimse yaptığı ödevi arkadaşına bakması için göndermesin.

Herkes kendisi yapsın selamlar. Benin gönderdiğim Tek yönlü liste c kodları şablonu örnek olarak kullanılabilir