Gai honetan ikasitako sarrera/irteerari buruzko kontzeptuak lantzeko, PCaren gailuetako batzuk oinarritzat hartuz (batez ere pantaila eta teklatua), laborategiko praktika batzuk proposatzen dira. Praktika hauetan landuko diren kontzeptuak hauek dira: memorian mapeatutako eta ez mapeatutako gailuen programazioa, inkesta bidezko sinkronizazioa, etenen bidezko sinkronizazioa, eten-kontroladorearen programazioa, eten bektorearen tratamendua, etab.

Praktika hauek egiteko C programazio lengoaia erabiliko da TurboC v3.0 plataformarekin. Makinaren hardwarea atzitzeko erabili behar diren funtzioak TurboC-ko liburutegietan oinarrituta daude eta programatuta ematen dira `makina.c` eta `makina.h` fitxategietan. Bestetik, praktika hauetan sortzen diren exekutagarriak probatu ahal izateko MS-DOS sistema eragilea behar da. MS-DOS sistema eragilea edo W98 sistema eragilea (MS-DOS sistema eragilea daukagu W98 azpian) instalatu ditzakegu gure makinan. Hala ere, guk Microsoft-en VirtualPC tresna (edo antzerako beste bat) erabiliz W98 alegiazko makina bat emulatuko dugu. TurboC erabiliz gure makinan programatutako praktikak alegiazko makinan exekutatu eta probatu ahal izateko bi makinen artean konpartitzen den karpeta bat erabiliko dugu.

Ondoren aurkezten dira egingo diren laborategiko 4 praktiken helburua zein den, praktikak egiteko erabiliko diren fitxategien antolaketa eta interesgarria izan daitekeen informazio gehiago.

1. praktika: pantaila testu moduan [panttest karpeta]
Praktika honetan memorian mapeatutako periferiko batentzat, PCaren pantailarentzat hain zuzen, zenbait funtzio programatuko dira. Aldi berean, pantailaren kurtsorearekin lan egingo dugu, kurtsorea memorian mapeatuta ez dagoen periferiko batek kontrolatzen duelarik.

Praktika hau aurrera eramateko erabiliko diren fitxategien antolaketari dagokionez, `makina.c` (eta berari dagokion `makina.h`) errutin moduluaz gain, `pantaila.c` (`pantaila.h`) eta `FrogaPantT.c` moduluak erabiltzen dira. `FrogaPantT.c` moduluan programa nagusia eta pantaila eta kurtsorearentzat programatu beharko diren errutinak probatzeko hainbat errutin aurki daitezke.

`pantaila.c` moduluan pantaila eta kurtsorearekin lan egiteko errutinak aurki ditzakegu. Hauetako errutin batzuk kodetuta daude eta beste batzuk, berriz, dokumentazioan azaltzen den periferikoaren funtzionamendu teorikoa kontuan hartuz programatu beharko dira. Proiektu guztia konpilatu ahal izateko `pantaila.h` fitxategian hau aurki dezakegu: hainbat definizio (`define` sasiagindua) eta errutinen buruak `extern` gako hitza aurretik dutela, beste konpilazio moduluetan erabili ahal izateko.

Kodetuta dauden errutinak hauek dira:

```
unsigned short PantHelb()
      //Kodetuta dagoen funtzio honek PCaren pantailaren memoriako oinarri helbidea
      //itzultzen du. Programa nagusiak funtzio honi deitzen dio eta jasotzen duen
      //oinarri-helbidea PantHas aldagaian gordetzen du

void IdatzTestu(int lerro, int zut, char * testu, unsigned char atrib)
      //Dagoeneko kodetuta dagoen funtzio honek testu bat idazten du pantailan
      //adierazitako lerro eta zutabetik aurrera
```

Kodetu beharreko funtzioak hauek dira:

```
void IdatzKar(int lerro, int zut, unsigned char kar, unsigned char atrib)
      //Kar karakterea pantailan idazten du, lerro eta zut parametroek
      //adierazitako posizioan (pantailako lerroa eta zutabea hurrenez hurren)
```

```
void PantEzabatu()
      //Pantaila guztia ezabatzen da, hau da, zuriuneak idazten dira pantaila osoan
      //zehar
void IrakurKarPant(int lerro, int zut, unsigned char *kar, unsigned char *atrib)
      //lerro eta zut-ek adierazitako pantailako posizioko karakterea irakurtzen du
void ScrollPant(int LerroKop)
      //Pantailaren scroll bat egiten du LerroKop lerrotakoa (pantailaren edukia
      //LerroKop lerro igotzen du
void JarriKurtsore(int x, int y)
      //Kurtsorea pantailako x, y posizioan jartzen du
void IrakurKurtsorePos(int *x, int *y)
      //Kurtsoarea zein posiziotan dagoen azaltzen du
void KurtsoreForma(int Hasi, int Buk)
      //Kurtsoreari forma ematen dio Hasi eta Buk arteko lerroak osatuz
      //karakterearen pixel matrizean
```

2. praktika: teklatua inkesta bidez [tekink karpeta]

Praktika honen helburua gailu baten inkesta bidezko sinkronizazioa programatzea da eta horretarako erabiliko den gailua PCaren teklatua da. PCaren teklatuaren sinkronizazioa etenen bidez izaten da eskuarki eta tekla bat sakatzean teklatuaren kontroladoreak IRQ1 lerroaren bidez etenak sortzen ditu. Praktika honetan, teklatuak IRQ1 lerrotik sortzen dituen etenak galaraziko ditugu eten-kontroladoreko IMR erregistroa atzituz eta inkesta bidezko sinkronizazioa nola egin ikusiko dugu. Inkesta bidezko sinkronizazioa aurrera eramateko, inkesta jarraitua egingo dugu eten-kontroladoreko IRR erregistroan: IRR erregistroko 1 bitak 1 balioa hartzen duenean tekla bat sakatu edo askatu dela jakingo dugu. Gogoratu horrelako inkestak programa nagusian egiten direla.

Praktika honetan, aurretik dauzkagun `makina.c` (`makina.h`) eta `pantaila.c` (`pantaila.h`) moduluetaz gain, `TekInk.c` (eta berari dagokion `TekInk.h` konpilazio banatua erabili ahal izateko) eta `FTekInk.c` moduluak dauzkagu. `FTekInk.c` moduluan programa nagusia eta inkesta egiten duen funtzioaren burua aurkitzen dira. Teklatuaren inkesta egiten duen funtzioa, `unsigned char TekInkestaIrakurri(),` kodetu behar da sakatu den teklaren ASCII kodea itzultzeko. Bestetik, `TekInk.c` moduluan honako funtzioak programatu behar dira:

```
unsigned char IRR_Irakurri()
      //IRR erregistroaren edukia itzultzen duen funtzioa
unsigned char TeklatuDatErregIrakurri()
      //Teklatuaren datu-erregistroaren edukia itzultzen duen funtzioa
void TeketenGalarazi()
      //Teklatuaren etenak maskaratzen dituen funtzioa. Hori egin bitartean eten
      //guztiakgalaraziko dira (IF=0) eta bukatu eta gero baimenduko dira berriz
      //(IF=1)
void TeketenBaimendu()
      //Teklatuaren etenak baimentzen dituen funtzioa. Hori egin bitartean etenak
      //guztiak galaraziko dira (IF=0) eta bukatu eta gero baimenduko dira berriz //
      //(IF=1)
void StrobeTeklatu()
      //Teklatuaren kontrol-erregistroaren 7. bitan STROBE sekuentzia egin
```

Era berean, suposatuko dugu jadanik definituta dagoela bektore bat, ASCII_TAULA[], non teklen posizio-kodea indize gisa hartuta, tekla bakoitzari dagokion ASCII kodea lor daitekeen, baina MAKE kasuan lortutako posizio-kodea bakarrik erabil daiteke indize gisa, ez BREAK kasukoa. Hau da, taula horretan begiratu dezakegu zein den tekla bati dagokion ASCII kodea tekla sakatzen denean, inoiz ez tekla askatzean.

3. praktika: teklatua etenen bidez [teketen karpeta]

Aipatu den bezala, PCaren teklatua etenen bidez kudeatzen da eskuarki eta eteteko eten-kontroladoreko IRQ1 lerroa erabiltzen du. Praktika honetan, tekla bat sakatzen denean exekutatuko den teklatuaren zerbitzu errutina sinple bat idatziko dugu. Guk idatzitako teklatuaren zerbitzu errutina hau exekuta dadin tekla bat sakatzean PCaren eten bektorean dagokion posizioa aldatu beharko dugu gure zerbitzu errutinaren helbidea gordez.

Praktika honek honela funtzionatuko du: programa nagusian PCa zerbait exekutatzen ari dela simulatzeko (`FTekEt.c` modulua) `DenbPasa` izeneko funtzioa exekutatuko da. `DenbPasa` funtzioaren exekuzioa `Amaitu` aldagai orokorrak 1 balioa hartzen duenean bukatuko da eta hori 'Q' tekla sakatzean gertatuko da. Tekla bat sakatzen denean, programa nagusia etengo da eta teklatuaren zerbitzu errutina `ZerErrTek` exekutatuko da. Zerbitzu errutina honek sakatu den teklaren ASCII kodea idazten du pantailan (eta sakatu den tekla 'Q' bada `Amaitu` aldagaiari 1 balioa ematen zaio).

Praktika honetan, `makina.c` (`makina.h`), `pantaila.c` (`pantaila.h`) eta `FTekEt.c` (programa nagusia) moduluez gain bi hauek dauzkagu: `TekEten.c` eta `EtenErr.c` (eta dagozkien `TekEten.h` eta `EtenErr.h`).

`TekEten.c` fitxategian teklatuaren kudeaketarekin lotutako funtzioak eta teklatuaren zerbitzu errutina daude:

```
unsigned char TeklatuDatErregIrakurri()
      //Teklatuaren datu-erregistroa irakurri eta itzuli.

void StrobeTeklatu()
      //Teklatuari STROBE bidali kontrol-erregistroko 7. bitean.

void IrakurKarTratatu(unsigned char c)
      //Kodetuta dagoen funtzio honek sakatutako tekla bere ASCII kodearen arabera
      //tratatzen du. Kurtsorea mugitzen du 'Y', 'G', 'B' edo 'H'teklak sakatzean eta
      //programa bukatuko da 'Q' tekla sakatzean.

void interrupt ZerErrTek()
      //Teklatuaren zerbitzu-errutina
```

`EtenErr.c` modulu edo fitxategian PCaren eten bektorea kudeatzen duten funtzioak daude. Programa nagusian, exekuzioaren hasieran, IRQ1 eten lerroari dagokion eten-bektoreko posizioa (9 sarrera) aldatzen da programatutako teklatuaren zerbitzu errutinaren (`ZerErrTek`) helbidea gordetzeko bertan. Eten-bektoreko posizio hori aldatu aurretik bertako edukia gordetzen da programa nagusia bukatzean eten-bektorea zegoen bezala utzi ahal izateko. Modulu honetan programatu beharreko funtzioak hauek dira:

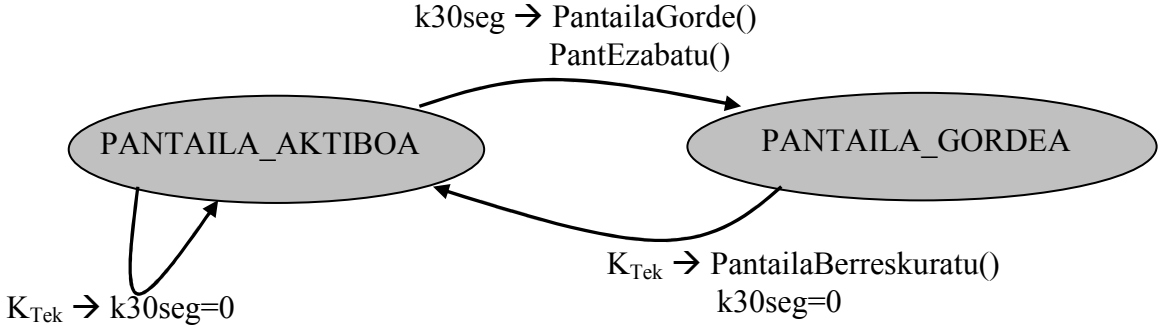
```
void AldatuEB(int sarrera, unsigned short IPBerria, unsigned short CSBerria,
              unsigned short * IPZah, unsigned short * CSZah)
      //Eten-bektorea aldatu "sarrera" (posizio zenbakia) adierazten duen osagaian
      //eta aurreko balioa gorde 4. eta 5. parametroetan

void BerreskuratuEB(int sarrera, unsigned short IPZah, unsigned short CSZah)
      //Eten-bektorean berreskuratuko dira hasieran "sarrera" osagaiak zituen balioak

void Eoi()
      //Etenen kontroladoreari Zerbitzu Errutina baten amaiera adierazteko.
```

Azkenik, programa nagusian eten-bektorea aldatu eta berreskuratzen duten funtzioei deiak aurkitzen dira. Dei hauetan parametroak falta dira: `AldatuEB(parametroak)` eta `BerreskuratuEB(parametroak)`.

4. praktika: erlojua eta teklatua [erlojua karpeta]

Praktika honen helburuak hauek dira: batetik, PCaren erlojua erabiliz denbora pasatzearekin lotuta dauden ekintzak nola programatu ikastea, eta bestetik, gailu bat baino gehiago, erlojua eta teklatua kasu honetan, tratatzen dituen praktika bat garatzea. Horretarako, pantaila salbatzaile sinple baten funtzionamendua programatuko dugu: 30 segundo pasatzen badira teklarik sakatu gabe, PCaren pantailan dagoena gorde behar da eta ez da berriro berreskuratuko tekla bat sakatzen den arte. Hortaz gain, segundoro ordua eguneratu behar da (`ORDUA` aldagaia da eguneratu behar dena; aldagai hau programa nagusian hasieratzen da) eta bi segundoan behin ordua pantailaratu beharko da. Programa exekutatzen hasten denetik minutu bat pasatzen denean programaren exekuzioa bukatu beharko da.

Irudian pantaila salbatzailearen funtzionamendua isaltzen duen automata ikus daiteke. Automatak bi egoera ditu: `PANTAILA_AKTIBOA` egoeran egoteak adierazten du ez direla 30 segundo pasa azken tekla sakatu zenetik, eta `PANTAILA_GORDEA` egoeran egoteak, berriz, 30 segundo pasa direla teklarik sakatu gabe eta pantaila gordeta aurkitzen dela. Automata zein egoeratan aurkitzen den kontrolatzeko `AutomataEgoera` aldagai globala daukagu. Bestetik, `k30seg` aldagai globala daukagu teklarik sakatu gabe 30 segundoak pasatu diren kontatzeko. Horretarako, automata `PANTAILA_AKTIBOA` egoeran aurkitzen bada erlojuaren zerbitzu errutinan `k30seg` aldagaia segundoro inkrementatu beharko da eta tekla bat sakatzen den bakoitzean teklatuaren zerbitzu errutinan aldagaia berriro hasieratu beharko da. `k30seg` aldagaiak 30 balioa hartzen duenean teklarik sakatu gabe 30 segundo pasa direla esan nahi du, eta une horretan erlojuaren zerbitzu errutinan pantailaren edukia gorde eta pantaila ezabatzen da automataren egoera aldatuz `PANTAILA_GORDEA` egoerara. Teklatuaren zerbitzu errutinaren ardura izango da `PANTAILA_GORDEA` egoeran teklaren bat sakatzen bada pantailaren edukia berreskuratzea.



Esan dugun bezala programaren exekuzioa bukatu beharko da minutu bat daramanean exekutatzen. Programari amaiera emateko eta lan horretan programa nagusia eta erlojuaren zerbitzu errutina sinkronizatzeko `Amaitu` aldagai globala erabiliko da. `Amaitu` aldagaiari 1 balioa eman behar zaio erlojuaren zerbitzu errutinan programak exekutatzen minutu bat daramala detektatzean. Une horretan automatikoki programa nagusiaren exekuzioa bukatuko da bertan begizta bat dagoelako `Amaitu` aldagaiaren balioa 0 den bitartean exekutatzen dena.

Proba honetan, programa nagusiaren hasieran eten-bektorearen bi sarrera edo posizio aldatzen dira, teklatuari dagokiona bat (9) eta erlojuari dagokiona bestea (0x1C, periferikoei buruzko teorian azaldu den bezala). Programa nagusiaren bukaeran eten-bektorea zegoen bezala uzteko bi posizio hauen balioak berreskuratu beharko dira.

Programa hau osatzen duten moduluei dagokionez, `makina.c` (`makina.h`), `pantaila.c` (`pantaila.h`) eta `EtenErr.c` (`EtenErr.h`) moduluez gain, hauek ere dauzkagu: `FErlojua.c` programa nagusia duen modulua, `TekEten.c` eta `Erlojua.c` (eta dagozkien `TekEten.h` eta `Erlojua.h`). `TekEten.c` moduluan teklatuaren kudeaketarako errutinak eta teklatuaren zerbitzu errutina dauzkagu:

```
unsigned char TeklatuDatErregIrakurri()
      //Teklatuaren datu-erregistroa irakurri eta itzuli.

void StrobeTeklatu()
      //Teklatuari STROBE bidali kontrol-erregistroko 7 bitean.

void interrupt ZerbErrTek()
      //Teklatuaren zerbitzu-errutina
```

`Erlojua.c` moduluan berriz, orduaren kudeaketarako errutinak, ordua pantailaratzeko errutina, pantailaren edukia gorde eta berreskuratzeko errutinak eta erlojuaren zerbitzu errutina dauzkagu.

```
void PantailaGorde()
      //Pantailaren edukia AurrekoPantaila aldagaian kopiatzen du

void PantailaBerreskuratu()
      //AurrekoPantaila aldagaian dagoena pantailan kopiatzen du pantailaren edukia
      //berreskuratuz

void OrduaEguneratu()
      //ORDUA aldagai orokorra eguneratzen du segundo batean

void OrduaIdatzi(int lerroa, int zutabea)
      //Adierazitako lerro eta zutabean ordua (ORDUA aldagai globala) pantailaratzen
      //du.

void interrupt ZerbErrErlojua()
      //Erlojuaren zerbitzu errutina.
```

Azkenik, praktika honetan ere, programa nagusian eten-bektorea aldatu eta berreskuratzen duten errutinei deietan (bai teklatu eta bai erlojurako) parametroak falta dira.