41. Fizička struktura n-arnog stabla i tehnika prošivki * Fizička struktura n-arnog stabla je u opštem slučaju obavezno spregnuta, a u posebnom slučaju može biti i sekvencijalna. * Deskriptor sadrži:1) indikator tipa strukture;2) ime stabla;3) adresu korena; 4) opis elemenata;5) red stabla. * Spregnuta struktura binarnog stabla može se izvesti primenom prošivki, tehnikom koja je vezana za operaciju obilaska. Ova tehnika na poziciji gde se nalazi prazan pokazivač upisuje poseban pokazivač (prošivku).

- prošivke na mestu desnih pokazivača predstavljaju adrese sledećih elemenata.

- ako su prazni levi pokazivači neiskorišćeni, varijanta nosi naziv Tehnika jednostrukih prošivki.

- ako se u neiskorišćene pokazivače upisuju adrese prethodnih elemenata u procesu obilaska, radi se o Tehnici dvostrukih prošivki.

42. Binarno stablo pristupa: definicija i osnovne operacije * Binarna stabla pristupa namenjena su prvenstveno za skraćenje postupka traženja, a da pri tome dodavanje i uklanjanje elementa budu i dalje mogući. * 1) pristup se obavlja na sledeći način:

- počinje se od korena; - ako je identifikator tekućeg jednak argumentu traženja, postupak je završen; - ako je taj identifikator veći prelazi se na levog;

- ako je manji prelazi se na desnog podređenog. 2) uklanjanje elemenata je dozvoljeno na svakoj poziciji, ali se pri tome red stabla ne sme menjati, kao ni ostale definicione osobine. 3) dodavanje elementa vrši se tako što se pri traženju mesta postojeće veze ne raskidaju, nego novi element postaje list.

43. Uopšteno stablo: definicija i osnovne operacije- * Glavna karakteristika uopštenog stabla je ta da je raspored podređenih elemenata proizvoljan i po potrebi promenljiv.* Jedna od definicija glasi:

Uopšteno stablo je uređeni par (x,p) gde je x-izdvojeni element, a p-jednostruko spregnuta lista čiji su elementi međusobno različita uopštena stabla. * Primitivne funkcije: koren, nadređeni, tekući.* Osnovne operacije:1) pristup – izvodi se tako što se zadaje kompletan put od korena ili tekućeg elementa do elementa kojem se pristupa;2) uklanjanje – može se ukloniti bilo koji element, mada se uglavnom uklanjaju listovi ili čitava podstabla;3) dodavanje – u praksi se vrši gotovo uvek na mestu lista.

44. Metode fizičke realizacije uopštenog stabla

* Fizička struktura stabla može biti: 1) prirodna fizička struktura;2) binarna fizička struktura. * Prirodna podrazumeva da svaki element stabla pored informacionog sadržaja ima još i k pokazivača (adresa) podređenih, kao i polje koje sadrži broj k. Nedostatak je promenljiv kapacitet memorijskog prostora zbog promenljivog broja pokazivača. * Binarna realizacija eliminiše problem promenljive dužine elemenata. Ostvaruje se tako što se podređeni elementi uređuju u jednostruko spregnutu listu, a nadređeni se povezuje sa prvim u listi.

45. Pojam tipa: Apstraktni i konkretni tipovi

* Ono što je u fazi projektovanja struktura podataka, to je u programskim jezicima tip.

* Apstraktni tip je logička struktura, a konkretni tip je fizička struktura +pitanje 46

46. Signatura i specifikacija tipa

* 1) Naziv tipa, 2) Primitivni tipovi (tipovi iz kojih se zadati tipovi izvode), 3) Sintaksa operacija (operacije koje se definišu za zadati tip),

4) Aksiome (i ograničenja).

* Uređena trojka (1,2,3) zove se signatura tipa.

Primer: - naziv tipa: tačka , - primitivni tipovi: real ,- sintaksa operacija: kreirati (x,y: real): tačka, apscisa (t: tačka):real,

ordinata (t: tačka): real,

rastojanje (t: tačka): real

* Uređena četvorka (1,2,3,4) je specifikacija tipa ili apstraktni tip podataka.

47. Generički tipovi - * Zovu se i prametrizovani tipovi. * Oslanjaju se na tipove koji nisu primitivni i promenljivi su

48. 1. Definicija strukturiranog programiranja - * Strukturirano programiranje je nastalo kao posledica "softverske krize" 60-tih godina. * Pod s. p. podrazumeva se skup tehnika za razvoj programskih modula koje koriste strogo definisane grupe upravljačkih struktura i struktura podataka. * Graf toka programa je digraf čiji čvorovi odgovaraju naredbama, njihovim delovima ili grupama. To je struktura slična blok dijagramu algoritma. Graf toka programa ima tri vrste čvorova: 1) proces ili funkcionalni čvor koji predstavlja operaciju obrade podataka ima ulazni i izlazni stepen 1.

2) predikat je čvor koji odgovara algoritamskom koraku logičkog testa. Ima ulazni stepen 1, a izlazni stepen 2

3) kolektor je čvor koji ne izvodi nikakvu obradu nego služi samo za povezivanje dve grane koje se stiču na istom mestu. Ima ulazni stepen 2, a izlazni 1.

* Pravilan program je program čiji graf toka zadovoljava uslove: 1) postoji samo jedna ulazna grana; 2) postoji samo jedna izlazna grana; 3) kroz svaki čvor prolazi najmanje jedan put od ulazne do izlazne grane.

* Prost program je program kome odgovara graf toka takaf da ni jedan od njegovih pravilnih potprograma nema više nego jedan čvor.

* Dva programa su ekvivalentna ako realizuju istu programsku funkciju, tj. ako za iste ulazne podatke daju isti rezultat.

49. Baza strukturiranih programa i osnovni prosti programi - * Skup prostih programa čijom se superpozicijom može realizovati bilo koji pravilan program nosi naziv bazba strukturiranih programa. * Strukturirani program definiše se kao program sastavljen od skupa prostih programa iz zadate baze. * Baze strukturiranih programa koje su od značaja svode se na proste programe sa jednim do četiri čvora. Takvih programa ima ukupno 15. Šest osnovnih prostih programa (u pascalu): 1) naredba dodeljivanja; 2) sekvenca begin-end; 3) if-then; 4) if-then-else; 5) while-do; 6) repeat-until.

50. Strukturna teorema i njen dokaz - * Teorema se bavi trima osnovnim strukturama: sekvencom, selekcijom (if-then-else) i iteracijom (while-do). Suština ove teoreme je da se svaki pravilan program može realizovati superpozicijom ova tri prosta programa. * 1) Ključni problem: dokaz da se prelazak sa jednog čvora grafa toka na sledeći (f-ja veze) može realizovati programskim segmentom koji ne sadrži ništa drugo osim sekvence begin-end i while-do. 2) označimo sve čvorove toka programa osim kolektora rednim brojevima:1,2,...

3) okolini pridružujemo oznaku 0;

4) funkcija veze definiše se pomoću kvadratne logičke matrice prelaza čijoj svakoj vrsti i koloni sa istom oznakom odgovara po jedan od čvorova grafa toka, uključujući i okolinu; 5) iz pravilnosti programa sledi da se u svakoj vrsti u trenutnku kada se dospe do odgovarajućeg čvora pri izvršavanju mora naći tačno jedan element sa vrednošću T, tako da je naredni čvor za izvršavanje određen jednoznačno.

51. Metoda Achroft-Manna za strukturiranje programa

* Metoda za prevođenje nestruktuiranog programa u struktuirani.

* Jedna je od najpoznatijih metoda i bazira se na srukturnoj teoremi. * Izvodi se tako što se (kao kod dokaza strukturne teoreme) čvorovi-procesi, čvorovi-predikati i okolina označe brojevima i uvede usmerivačka promenljiva, a zatim se odgovarajući graf toka programa prevede u oblik koji je strukturiran mehanizmom identičnim onim kod dokaza teoreme.

52. Metoda sukcesivne dekompozicije - * Ova metoda se koristi za samu izradu programa, s ciljem da primarni oblik programa bude struktuiran, i da se ne zahtevaju naknadne intervencije. * Suština metode je postepen prelaz od opšteg opisa programa ka njegovoj realizaciji, organizovan tako da se u fazama vrši sve veća detaljizacija. * Ovaj metod ima svoje prednosti i nedostatke: - Prednosti su: 1) nezavisna razrada delova programa eliminiše skokove, čini kod razumljivijim smanjujući time verovatnoću greške, omogućava timski rad na projektu; 2) timovi

produkuju module koji su slični po stilu pa se lako mogu povezati u celinu.

- Nedostaci: 1) može se desiti da se strukture podataka kojima operišu moduli međusobno razlikuju, pa je module teško spojiti u celinu; 2) problem složenih softverskih sistema: za složene sisteme veoma je teško ako ne i nemoguće definisati postupke na visokim nivoima dekompozicije (operativni sistem); 3) ranije napisani delovi programa ne mogu se kasnije koristiti jer program razvijen sukcesivnom dekompozicijom je kompaktna celina.

53. Faze životnog ciklusa programa i osnovni pristup programiranju - * Kaskadni model životnog ciklusa programa: 1) analiza (sistem analize); 2) projektovanje; 3) kodiranje(fizička realizacija); 4) testiranje; 5) eksploatacija; 6) modifikacija.

Postoje posebne metode za sve ove stavke u životnom ciklusu.

* Struktuirana tehnologija prati ceo životni ciklus. Danas je dominantna objektna tehnologija. “Objektno programiranje je kreativna negacija struktuiranog”.

54. Osnovni pojmovi u objektnom programiranju: objektno projektovanje, objektno programiranje, klasa i objekat, osobine objekata.- * Objektno orijentisano programiranje (OOP) je nov način razmišljanja u oblasti razvoja softvera.

* Objekat (instanca) je deo memorije nekog izvršnog programa koji poštuje korisnički definisana pravila. Objekat sadrži karakteristike entiteta (atribute) i njihovo ponašanje (metode).

* Klasa je korisnički definisan tip podataka za kreiranje objekata. Objekat je instanca klase. Klasom modelujemo entitete.

* Atributi (podaci članovi) su bitne karakteristike entiteta. Metode (funkcije članice) su operacije nad entitetima. * Objekti se karakterišu: 1) identitetom (imenom); 2) stanjem (koje je definisano trenutnim vrednostima atributa); 3) ponašanjem (metodama). * Sredstva objekne tehnologije su:

1) apstrakcija;2) modularizacija i inkapsulacija;3) nasleđivanje;4) polimorfizam.

* Apstrakcija je skrivanje informacija. Fizički objekti često imaju spoljašnji oklop koji krije unutrašnju složenost. * Nasleđivanja – proširivanjem objekata mogu da se dobiju novi objekti.

Kod nasleđivanja kažemo da je jedan objekat izveden iz drugog.* Polimorfizam – različiti objekti mogu da rade na isti način.

55. Specifikacija objekata u Turbo Paskalu - * U konceptu objektno orijentisanog programiranja podaci (atributi) i procedure se kombinuju u objekte. Objekat sadrži karakteristike entiteta (podatke) i njihovo ponašanje (procedure). Spajanjem ove dve stvari objekat zna sve što mu je potrebno da završi određeni posao. * Procedure i funkcije koje su deklarisane u okviru objekta nazivaju se metodama. Objekat definiše samo zaglavlje za metodu, a kod se specificira odvojeno. * Objekat može da se smatra slogom koji sadrži polja sa podacima i deklaracije metoda. * Nakon što je objekat definisan, svim akcijama koje se odnose na njega, može se pristupiti referisanju na sam objekat.

56. Inkapsulacija objekata u Turbo Paskalu - * Enkapsulacija (upakivanje) je jedan od najvažnijih ciljeva objektnog programiranja. Ona predstavlja kreiranje objekata koji funkcionišu kao potpuni entiteti. Jedno od pravila enkapsulacije je da programer ne mora direktno pristupati podacima koji se nalaze u okviru objekta. Umesto toga, u okviru objekta se definiše metoda koja manipuliše podacima. To je osnovna prednost enkapsulacije – pristup podacima indirektno uz pomoć metoda. * Jedan od načina da se postigne enkapsulacija je primena modula.

Deklaracija objekta se postavlja u okviru
Interface
sekcije, a tela metode se postavljaju u okviru
Implementation
sekcije. U okviru jednog modula može se deklarisati proizvoljan broj objekata, ali se obično definiše samo jedan.

* Pomoću rezervisane reči *Private* deklaracija objekata se deli na "javni prostor" i "privatni prostor". Javni prostor sadrži podatke i metode koje su dostupne i ostalim modulima. Privatni prostor sadrži podatke i metode koje su dostupne samo potprogramima u okviru istog modula. * Apstrakcija je skrivanje informacija. * Modularnost je razbijanje programa

na celine. U TP-u moduli su *Unit*-i. * Inkapsulacija je tehnika objedinjavanja strukture i ponašanje klase u softversku celinu, takva da realizacija bude nedostupna klijentu. * Modul je softverska komponenta koja se realizuje (projektuje, programira, testira) autonomno.

57. Nasleđivanje i njegove karakteristike - * Objekti osim što sadrže sopstvene podatke i metode, mogu i da ih naslede od drugih objekata.

Primer: *type*

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□.

□□□□□□□□□□□□*Osoba = Object*

□□□□□□□□□□□□.

□□□□□□□□□□□□.

□□□□□□□□□□□□*Student = Object(Osoba)*

Objekat *Student* nasleđuje sve što sadrži objekat *Osoba,* uključujući podatke i metode. Objekat *Osoba*
je predak, a objekat
Student
je potomak. Objekat može imati više od jednog pretka.

* Nasleđivanjem metoda izbegava se pisanje iste metode koja je potrebna za dva objekta.

* Kada se u okviru objekta definiše naziv polja sa podacima, nijedno drugo polje u bilo kom objektu potomku ne može imati isti naziv. *

Nasleđivanje je preuzimanje u datu klasu atributa i metoda iz druge klase uz mogućnost modifikovanja metoda.

58. Polimorfizam objekata - * Polimorfizam je mogućnost procedure da prihvati više objektnih tipova, čak iako nisu definisani prilikom prevođenja. * Modul koji sadrži rutine koje prihvataju polimorfne promenljive, moguće je prevesti i u takvom obliku distribuirati bez izvornog koda. U tom slučaju će korisnici biti u mogućnosti da kreiraju sopstvene objekte koji funkcionišu u kombinaciji sa prevedenim procedurama. * Prava vrednost polimorfizma dolazi do izražaja kada se uključe i virtuelne metode i dinamički objekti.

* Polimorfizam je osobina objekata, takva da različiti objekti mogu da rade na isti način.

* Postoje tri tipa: 1) preklapanje poruka (višeznačnost poruka);

2) polimorfizam objekata kao stvarnih parametara;

3) polimorfizam operatora dodele vrednosti.

59. Statički i dinamički tipovi - * Statički tip je tip koji je deklarisan. * Dinamički tip je promenljiv i on je rezultat dodele vrednosti. Dinamčki tip je uvek potomak statičkog tipa.

60. Statičke i virtuelne metode u Turbo Paskalu - * Statičke metode se izvršavaju gotovo identično kao i obične procedure i funkcije. Pozivi ovih metoda se povezuju u toku prevođenja. Ovaj proces se zove rano povezivanje. * Virtuelne metode – povezivanje se vrši u toku izvršavanja tek nakon poziva metode. Ovaj proces se naziva kasno povezivanje. * Postoje dve koraka za pretvaranje statičke metode u virtuelnu: 1) kreiranje konstruktor metode, koja ukazuje da će za taj objekat biti korišćene virtuelne metode. Da bi se kreirao konstruktor, rezervisanu reč *Procedure* menjamo sa *Con structor*
i u okviru deklaracije objekta i u definiciji metode.

2) zaglavlju metode u okviru deklaracije objekta treba dodati
Virtual.
* Postoje tri pravila kod virtuelnih metoda:

1) obavezno se mora pozvati konstruktorska metoda pre poziva bilo koje druge virtuelne metode. 2) kada se metoda deklariše kao virtuelna, onda ona i u potomcima mora biti deklarisana kao virtuelna.
3) kada je metoda deklarisana njeno zaglavlje se ne može promeniti u okviru nijednog potomka.

61. Definicija specifikacije, korektnog programa, arene i test skupa - * Testiranje programa je jedna od najskupljih i najvažnijih aktivnosti u životnom ciklusu programa. Testiranjem se utvrđuje u kojoj meri program obavlja posao za koji je namenjen i kako se ponaša u različitim eksploatacionim uslovima. * Specifikacija je pravilo koje propisuje način reakcije programa na različite ulazne podatke. * Program P je korektan ako važi: *dom([P]□ [S]) = dom([S])* tj. ako se za svaki ulaz program zaustavlja i ako se za svaki ulaz dobija izlaz predviđen specifikacijom. * Uređena trojka (P,S,D) je arena, i ona oslikava ambijent u kom se izvodi testiranje. P-program, S-specifikacija, D-domen. * Test skupa T(Test Set) je svaki konačan podskup skupa D, tj. za neki ulaz

x Є T
može da važi:

1) Za ulaz x program se ponaša korektno *(x,[P](x)) Є [S];*□ 2) Za ulaz x program se ne ponaša korektno *(x,[P](x))ne Є[S];* 3) Vrednost
(x,[P](x))
nije obuhvaćena specifikacijom (pogrešan izlaz); 4)
(x,[P](x))
nije definisano (program se za vrednost x ne zaustavlja).

62. Testiranje programa bez primene računara - * Testiranje programa bez primene računara ili analiza programa je jedna od strategija destruktivnog testiranja, koje podrazumeva analizu izvršavanja programa sa ciljem otkrivanja grešaka. * Analiza programa obuhvata dve metode: 1) analiza programskih segmenata 2) simulacija izvršavanja programa. * Prednosti ove metode: 1) mogućnost otkrivanja 30-70% grešaka; 2) osim što se utvrđuju greške, one se i lociraju; 3) ova strategija je prilično jeftina.

* Nedostatak ove metode je taj što je izvode ljudi pa je teško otkriti suptilne greške koje predstavljaju najveći problem.

* Analiza programskih segmenata: - koristi se za testiranje modula; - vrši je tim od 4 osobe: moderator, autor programa, i dva člana, specijalisti za izradu softvera; - testiranje traje 90-120 minuta (150 naredbi na sat). * Simulacija izvršavanja programa:

- koristi se za testiranje veza među pojedinim modulima; - tim se sastoji od 3-5 osoba: moderator, sekretar, izvođač testa i dva iskusna programera (jedan član je autor programa); - testiranje traje 1-2 sata.

63. Metoda bele kutije za testiranje - * Ova metoda podrazumeva detaljno poznavanje koda jer se test podaci projektuju na bazi izvornog koda. * Neke od metoda ove strategije su: 1) prekrivanje naredbi; 2) prekrivanje odluka; 3) prekrivanje uslova; 4) prekrivanje odluka-uslova; 5) prekrivanje višestrukih uslova. * Metoda prekrivanja naredbi predviđa definisanje takvih ulaznih podataka koji obezbeđuju da se svaka naredba programa izvrši bar jednom.

* Metoda prekrivanja odluka predviđa definisanja ulaznih podataka takvih da za svaki algoritamski korak odluke bar jednom budu realizovana oba ishoda (da, ne). * Metoda prekrivanja uslova je alternativna tehnika za prekrivanje odluka. Pod uslovom se podrazumeva svaki potpredikat u odlukama. * Metoda prekrivanja odluka-uslova zahteva veći broj ulaza. Ulazni podaci se podešavaju tako da svaka odluka ima oba ishoda i svaki uslov u okrivu odluke ima oba ishoda. * Metoda prekrivanja višestrukih uslova je tehnika kod koje se ulaznim podacima izaziva prekrivanje svih kombinacija predikata u svim odlukama.

64.. Metoda klasa sličnosti za testiranje (crna kutija 1/2) - * Metoda klasa sličnosti za testiranje je jedna od metoda strategije crne kutije, gde se ne konsultuje izvorni program, nego se test podaci kreiraju na osnovu specifikacije programa. Program se posmatra kao crna kutija čiji detalji nisu od interesa za testiranje. * Ova metoda spada među najjednostavnije među metodama crne kutije i veoma se često koristi u praksi. * Ideja je da se skup mogućih ulaza podeli na klase sličnosti. Cilj je da se izabere samo jedan član cele klase, čime se broj test ulaza smanjuje. Postupak ima dve faze: 1) U prvoj se identifikuju klase sličnosti za validne i invalidne ulaze. Identifikacija se vrši na osnovu specifikacije i iskustva.2) Predstavnici definisanih klasa sličnosti se kombinuju u test skup po pravilima: validne klase se kombinuju u što manji broj test ulaza, a za svaku invalidnu klasu formira se poseban test ulaz. * Ova metoda ne propisuje koji ulaz iz klase sličnosti treba izabrati za test.

65. Metoda graničnih vrednosti za testiranje (crna kutija 2/2) - * Metoda klasa sličnosti za testiranje je jedna od metoda strategije crne kutije, gde se ne konsultuje izvorni program, nego

se test podaci kreiraju na osnovu specifikacije programa. Program se posmatra kao crna kutija čiji detalji nisu od interesa za testiranje. * Ova metoda uzima u obzir činjenicu da ako program ima grešaka one se ispoljavaju u graničnim slučajevima. * Metoda je slična metodi klasa sličnosti, ali ima dve modifikacije: 1) Kod izbora test ulaza iz klasa sličnosti uzimaju se predstavnici koji su neposredno uz granice koje razdvajaju validne od invalidnih klasa. 2) Formiraju se ulazi vezani za izlazne podatke

66. Metoda simboličkog izvršavanja za testiranje - * Metoda simboličkog izvršavanja za testiranje je jedna od metoda konstruktivnog testiranja i ima cilj da pruži dokaz da program nema grešaka. * Ova metoda je najstarija i najpoznatija za testiranje manjih programa i delova većih programa. * Postupak se odvija tako što se ulaznim promenljivama umesto konkretnih dodeljuju simboličke vrednosti, a zatim se simulacijom, bez računara ili primenom odgovarajućih programa odredjuju simboličke formule za izlaze koji se na kraju uporedjuju sa specifikacijom. * Nedostaci ove metode: javljaju se problemi sa naredbama tipa *IF* i sa ciklusima, jer se moraju uvoditi dodatne pretpostavke da bi bili predviženi različiti ishodi. Kod
IF
–
THEN
–
ELSE
oba ishoda, a kod ciklusa i nastavak i izlazak iz ciklusa.

67. Testiranje složenih programa - * Kod testiranja složenih programa glavni problem je redosled testiranja pojedinih modula. Ya testiranje svakog modula potrebno je imati na raspolaganju: 1) module iz kojih se poziva testirani modul; 2) module koje testirani modul poziva. * Za testiranje modula potrebno je napisati pod module, gde treba voditi računa da broj podmodula bude što manji, a da se izbegne testiranje celog programa. * Postupak se može skratiti teko što se već istestirani moduli mogu koristiti za testiranje njemu nadređenih i podređenih. Metode koje koriste ovu pogodnost zovu se inkrementalna testiranja. * Postoje dve metode za inkrementalno testiranje: 1) testiranje s vrha ka dnu – gde se počinje od glavnog modula; 2) testiranje sa dna ka vrhu – gde se polazi od modula najnižeg nivoa.

68. Principi izrade i funkcionisanja korisničkog interfejsa - * Korisnički interfejs je deo programa namenjen za razmenu podataka između programa i korisnika. * Osnovna hardverska sredstva za izradu k.i. su: ekran, tasteri na tastaturi, miš, zvučnik... Kombinovanjem ovih sredstava, uz uvažavanje određenih konvencija može se ostvariti dobar korisnički interfejs. * Principi kojima se treba rukovoditi prilikom izrade KI su: 1) u svakom momentu korisnik mora da zna stanje programa; 2) ekran treba da sadrži što više informacija o trenutno aktivnim funkcijama; 3) ekran ne sme biti pretrpan informacijama; 4) mora postojati help sistem; 5) ista poruka treba da se pojavljuje na istom mestu.

69. Elementi korisničkog interfejsa: desktop, horizontalni i vertikalni meni, statusna linija, prozor, okviri, softverski tasteri. * To su najčešće korišćeni elementi korisničkog interfejsa. * Desktop je statički definisana nepromenljiva površina ekrana na kojoj se nalaze ostali elementi interfejsa. * Horizontalni meni omogućava kontrolisani pristup glavnim funkcijama programa. Nalazi se na vrhu ekrana i nepromenljiv je. sastoji se od: dve ili više stavki, markera koji označava aktuelnu stavku, i skraćenica (shortcut keys). * Vertikalni meni podseć na horizontalni, jer se sastoji od stavki, ima marker i skraćenice. Razlika je u tome što se on dinamički pojavljuje na ekranu i po obavljenom poslu nestaje. Najčešće je izveden kao podmeni horizontalnog ili podmeni drugog vertikalnog menija. * Statusna linija ima zadatak da obezbedi najbrži pristup do glavnih funkcija programa. Taj pristup se ostvaruje aktivnim tasterima (hotkeys) čiji opis stoji na liniji. * Prozori služe za definisanje radne zone korisnika, tj. zone u kojoj se dešava interakcija između programa i korisnika. * Okviri predstavljaju sklopove koji se prave od osnovnih delova u koje spadaju softverski tasteri, text, labele, ulazne linije... * Softverski tasteri su slični aktivnim tasterima s tastature, tj. pritiskom

70. Upravljanje ekranom i tastaturom u TP

* Upravljanje ekranom vrši se posredstvom video memorije. To je logički deo ukupne operativne memorije koji je hardverski rešen tako da upis koda u nju izaziva ispisivanje određenih simbola na ekranu. Svakom znaku na ekranu odgovara 2 bajta video memorije, niži bajt sadrži ASCII kod znaka , a viši atribute boje. U pascalu je najbolje koristiti specijalni atribut *absolute* koji se može vezati za svaku promenljivu.

* Rukovanje tastaturom obavlja se pomoću dve funkcije: ReadKey i KeyPressed.

Pritisak na taster izaziva upis odgovarajuće informacije u poseban cirkularni red koji je deo sistemske memorije operativnog sistema. Ako je pritisnut neki od tastera: slovo, broj ili specijalni znak, u cirkularni red se upisuje jedan bajt koji sadrži ascii kod znaka. Ako je pritisnut neki od funkcijskih tastera, ili neka kombinacija (alt, ctrl, shift)+taster, u cirkularni red će biti upisana dva bajta. Prvi uvek ima vrednost 0, a drugi ima kod tastera ili kombinacije tastera(scan code).

71. Principi rukovanja bojom i rukovanje bojom u TP

*Bojom znaka na ekranu manipuliše se pomoću višeg od dva bajta koji pripadaju znaku (bajt atribut). On sadrži informacije o boji teksta, boji pozadine, intenzitetu i blinkovanju.

* Svaka boja dobija se kombinacijom tri osnovne: crvene, zelene i plave, koje čine RGB paletu.

* Pascal omogućava rukovanje bojama putem dve procedure TextColor i TextBackground koje se nalaze u CRT jedinici.

* Procedura TextColor služi za izbor boje znaka, intenzitet i blink. Ona postavlja boju texta na zadatu vrednost i ta boja se zadržava dok se ne promeni ponovnim pozivom te procedure.

* Isto važi za proceduru TextBackground, s tim da se boja odnosi na pozadinu.

72. Principi rukovanja zvukom i rukovanje zvukom u TP

* Zvuk se upotrebljava u dve svrhe: 1) opomena – pišteći zvuk kada program dođe u neželjeno stanje; 2) afirmacija – konstantan zvuk kad se neki važan posao završi

* Opominjući zvuk, beep, dobija se primenom naredbi write (chr(7)) ili write (#7)

* Pascal predviđa još tri procedure za rukovanje zvukom: 1) Sound(hz:word) – generiše zvuk učestanosti zadate parametrom hz, sa neograničenim trajanjem; 2) Delay (ms:word) – izaziva čekanje u trajanju od ms milisekundi; 3) NoSound – prekida se zvuk.