Univerza v Mariboru,
Fakulteta za elektrotehniko, računalništvo in informatiko

RAČUNALNIŠTVO
1. letnik VSŠ

PRIPRAVA
RAČUNALNIŠTVO

Maribor, 2012

Univerza v Mariboru,
Fakulteta za elektrotehniko, računalništvo in informatiko

RAČUNALNIŠTVO
1. letnik VSŠ

Priprava

Računalništvo

VPRAŠANJA IN ODGOVORI ZA

2. KOLOKVIJ

Avtor:
Gregor Nikolić

Maribor, 2012

KAZALO

KAZALO SLIK

1. UVOD

Priprava obsega vprašanja in odgovore za drugi kolokvij pri predmetu Računalništvo, smer študija Elektrotehnika VS na Univerza v Mariboru, Fakulteta za Elektrotehniko Računalništvo in informatiko.

2. VPRAŠANJA

2.1 Programski jezik C

1. Narišite primer diagrama poteka za while/do while/for/if stavek (št. točk: 6)
2. Na podlagi podanega diagrama poteka napišite program v C-ju (opomba: poudarek bo na if, while, do while in for stavku) (št. točk: 6)
3. Na kakšne načine lahko določamo tip podatkov (za znake, za cela števila in za realna števila)?
4. Napišite primer določitve za konstanto (PI=3.14159)!
5. Kakšna je razlika med avtomatičnimi in zahtevanimi pretvorbami med tipi? Napišite primer!
6. Kakšne so razlike med globalnimi in lokalnimi spremenljivkami? Pokažite na primeru!
7. V katerem primeru spremenljivka obdrži vrednost tudi po izhodu iz funkcije in zakaj je to potrebno?
8. Kaj so oštevilčeni tipi? Napišite primer za "visoko, srednjo, nizko temperaturo"!
9. Kaj so funkcije in zakaj jih uporabljamo?
10. Opišite strukturo funkcije!
11. Opiši deklaracijo in definicijo funkcije!
12. Napišite konkreten primer funkcije v programskem jeziku (void/vrne eno število/vrne več števil)!
13. Kakšne so prednosti in slabosti uporabe funkcij?

2.2 Vhodi/Izhodi funkcije

1. Možno vprašanje A: Koliko vhodov in izhodov imajo funkcije?
2. Možno vprašanje B: imajo funkcije vhodne/izhodne spremenljivke in kako jih realiziramo?
3. Na katere tri načine lahko uporabimo funkcije v programu?
4. Kaj pomeni izraz "zaglavje" v C programu in kako jih uporabljamo?
5. Kaj so kazalci in kako jih uporabljamo ter predstavljamo? Kaj so vrednosti, naslov in vsebina kazalca? Pokažite na primeru!
6. Kako uporabimo kazalce, da dobimo izhodne spremenljivke funkcij?
7. Ali lahko uporabimo kazalce na funkcije? Navedite primer!
8. Opišite vektorje in polja v programskem jeziku C!
9. Opišite strukture v programskem jeziku C!
10. Kako definiramo polje znakov v programskem jeziku C? Koliko znakov je v resnici v polju znakov in opišite zakaj je razlika?
11. Navedite vsaj štiri najbolj pogosto uporabljene specifikacije (za %) za formatirano branje in pisanje!
12. Ali lahko dinamično dodeljujemo pomnilnik in kako?
13. Bitne operacije - kako je izvedeno branje enega bita?
14. Bitne operacije - kako je izvedeno brisanje enega bita?
15. Bitne operacije - kako je izvedeno postavljanje enega bita?

2.3 Postopki pri prevajanju programov

1. Kaj vse potrebujemo za izvedbo postopka prevajanja?
2. Kaj so prevajalnik, povezovalnik, knjižnice in generator binarne kode?
3. Kaj so mnemoniki pri asemblerju in zakaj jih uporabljamo?
4. Opišite potek prevajanja!

5. Kako poteka prevajanje enostavnega in bolj zapletenega programa?
6. Kaj je razhroščevalnik (debugger) in kako deluje?
7. Kako najdemo napake v programu s pomočjo razhorščevalnika (debuggerja)?
8. Kaj je simulator in kako deluje? Navedite primer!

2.4 Pristopi h kvalitetnemu programiranju

1. Kakšne značilnosti ima kvalitetna programska oprema?
2. Katero je zlato pravilo razvoja programske kode? Utemelji odgovor!
3. Katere so pet zapovedi za uspeh pri delu?
4. Katere glavne faze razvoja programa poznamo?
5. Kaj in koga analiziramo na začetku razvoja programa?
6. Opišite kako in zakaj uporabljamo ocenjevanje?
7. Kako deluje top-down/(buttom up) metoda?
8. Kako lahko poimenujemo besede pri programiranju in na kaj moramo paziti pri poimenovanju?
9. Kaj se zgodi, če delimo z 0? Utemeljite odgovor?
10. Kako naredimo program "berljiv"?
11. Katere so najbolj pogoste napake pri pisanju programov?
12. Naštejte in natančno opišite vsaj štiri pravila kvalitetnega programiranja!
13. Ali je pravilno glede na zahteve po kvalitetnem programiranju (opomba: podano na izpitu/testu)?
14. Kakšna je razlika med strukturnim in objektnim programiranjem?

2.5 Računalniški sistem

1. Opišite osnovno zgradbo računalniškega sistema
2. Natančno opišite in narišite osnovno zgradbo računalnika in nato opišite, kako deluje centralna procesna enota (CPU)!
3. Kaj je cikel procesiranja? Kako merimo njegovo hitrost in kateri so faktorji, ki vplivajo na hitrost?
4. Kaj sta aritmetično logična enota in krmilna enota ter kako delujeta?
5. Kaj je register in kako deluje?
6. Kaj so dual-core in multi-core procesorji?
7. Kaj so hladilniki?
8. Kaj je bit, nibble, byte?
9. Kako merimo količino podatkov in kako hitrost prenosa podatkov? Kako predstavljajo kilobyte, megabyte, gigabyte, terabyte in petabyte?
10. Kakšne vrste pomnilnikov poznamo in kako jih razvrščamo?
11. Kaj so razširitvene reže in vmesniške kartice?
12. Opišite in obrazložite vsaj pet vrat in konektorjev (ports and connectors)?
13. Kaj so USB/Firewire/DVI/HDMIi/S-video konektorji in kje jih uporabljamo?
14. Na kaj vse moramo paziti, ko kupujemo računalnik?
15. Opišite vrste računalnikov!

2.6 Vhodno/izhodne enote

1. Naštejte in natančneje opišite vsaj tri vhodne naprave!
2. Naštejte in natančneje opišite vsaj tri izhodne naprave!
3. Katere biometrične vhodne naprave poznamo?
4. Kateri so problemi pri uvajanju biometričnih naprav?
5. Kaj so tipkovnice, kako jih delimo in kakšna je razlika med QWERTY in QWERTZ tipkovnicami? Katere druge tipe tipkovnic še poznamo?
6. Kaj so lokatorji in posebej opišite vsaj tri lokatorje!

7. Katere vrste zaslonov najpogosteje uporabljamo?
8. Kakšna je razlika med prikazovalniki, kot so: (Opomba: podani bodo na izpitu)?
9. Opišite 3D interaktivno animacija?
10. Opišite zakaj nastopajo še vedno problemi pri zaznavi človeškega govora in kako se danes zajema govor v računalnik?
11. Kaj je skener in zakaj ga lahko uporabljamo?
12. Kaj je MIDI zvočni sintetizator ter kako in kje ga uporabljamo?
13. Kaj so senzorji in katere senzorje uporabljamo v elektrotehniki?
14. Kaj so force feedback naprave? Navedite primer njihove uporabe?
15. Kakšna je razlika med analognimi in digitalnimi signali?
16. Opišite vsaj tri prikazovalnike za avtomatizacijo procesov!
17. Kako zajemamo zvok/video?
18. Katere standarde poznamo za video obdelavo in zakaj potrebujemo kompresirani video?
19. Kakšna je razlika med DLP/CRT/OLED/SEP/LCD/plasma zasloni?

2.7 Programska oprema

1. Narišite in opišite skico razdelitve programske opreme!
2. Naštejte in opišite vsaj štiri primere programov za: poslovne II grafične in multimedijske II izobraževalne II komunikacijske II pomožne programe!
3. Kakšne so razlike med licenčnimi programi, odprto kodnimi programi, shareware, freeware in public domain programi?
4. Naštejte in opišite vsaj štiri programe za vzdrževanje računalnika in integrirana okolja!
5. Kakšna je vloga programov za urejanje dokumentov in kaj nudijo?
6. Naštejte in opišite vrste programov za urejanje dokumentov!
7. Kakšna je razlika med editorji II urejevalniki besedil II programi za namizno založništvo II oblikovalci besedil II orodji za tvorbo spletnih strani ?
8. Kakšna je vloga in funkcija programov za delo s tabelami?
9. Kakšna je vloga in funkcija programov za delo s podatkovnimi bazami?
10. Kaj pomeni izraz "relacija" na področju podatkovnih baz in kako jo uporabljamo?
11. Katere jezike in programe uporabljamo za podatkovne baze? Kaj je njihova značilnost?
12. Kako se delijo enote v podatkovnih bazah? Opišite te enote in navedite primer za vsako enoto!
13. Kakšna je vloga in funkcija programov za vstavljanje opomb! Navedite primere!
14. Kakšna je vloga in funkcija programov za prezentacije! Navedite primere!
15. Kakšna je vloga in funkcija programov za matematiko! Navedite primere!
16. Kakšna je vloga in funkcija programov za risanje! Navedite primere!
17. Kakšna je razlika med bitno in vektorsko sliko?
18. Kakšna je vloga in funkcija programov za zvočno zajemanje! Navedite primere!
19. Kje so problemi pri zvočnem zajemanju govora? Katera dva načina za zaznavanje sposobnosti izgovorjene besede poznamo?
20. Kakšna je vloga in funkcija programov za video zajemanje! Navedite primere!
21. Kakšna je vloga in funkcija programov za projektno vodenje! Navedite primere!
22. Kakšna je vloga in funkcija programov za osebno informiranje! Navedite primere!
23. Navedite in opišite vsaj štiri internetna orodja!
24. Kako naredimo 3D fotografske slike?
25. Kako naredimo panoramske fotografske slike?
26. Na kaj vse moramo paziti pri nakupu programske opreme?
27. Kaj pomenijo izrazi: PIM / CAD / SQL / Open Source / HTML / XML in kje ter v kakšne namene jih uporabljamo?

2.8 Vprašanja in odgovori iz kvizov za samostojno preverjanje znanja

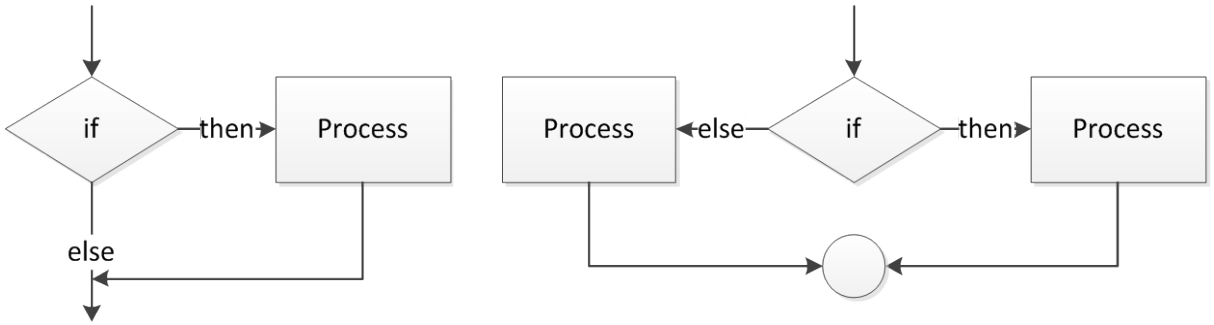
1. Programiranje v programskem jeziku C
2. Kateri od predznakov se ne uporablja za kazalce?
3. Funkcijo določimo na začetku na naslednji način: function (a,b); .
4. vstavi manjkajočo besedo) Programski modul v jeziku C se imenuje (v originalu) _____ .
5. Strukturo definiramo s stavkom “structi Osebni_podatki oseba”.
6. Z “enum” ukazom izdelamo oštevičeni tip spremenljivk.
7. Kateri izmed izrazov ni polje ali matrika?
8. S kazalci ne moremo kazati na funkcije.
9. Polje znakov na koncu besede vsebuje znak “\0” .
10. (vstavi manjkajočo besedo) Izraz _____ se uporablja zato, da povemo, da funkcija ne vrne nobene vrednosti.
11. Osebni računalnik (strojna oprema)
12. Katera naprava ne spada med vhodne naprave?
13. Koliko je 1 Kbyte?
14. Super računalniki so sposobni dosegati hitrosti večje od?
15. Kateri od navedenih ne spada med vrste notranjih pomnilnikov?
16. Kaj dela aritmetično logična enota?
17. Katera naprava ne spada med izhodne naprave?
18. Kateri stavek kaže na pravilni vrstni red glede hitrosti pomnilnikov?
19. Kateri priključek ne spada med priključke za zaslon?
20. Kaj so in kako delujejo "dual-core" procesorji?
21. Kaj ni del centralno procesne enote?
22. Osebni računalnik (programska oprema)
23. Licenčne programe lahko uporabljamo brez plačila avtorju izdelka.
24. Kateri program ne spada med internetna orodja?
25. Program za tehnično risanje (CAD) omogoča risanje slik v vektorski obliki.
26. Programi za projektno vodenje so namenjeni planiranju in spremljanju aktivnosti.
27. Katera funkcija ne spada v programe za urejanje dokumentov?
28. Kateri program uporablja bitno sliko?
29. Kateri način ni pravilen za zvočno zajemanje in razumevanje izgovorjene besede?
30. Kaj ne spada v program za osebno informiranje?
31. Kaj so programi za vzdrževanje računalnika in integrirana okolja?
32. Kaj so baze podatkov?

3. VPRAŠANJA IN ODGOVORI

3.1 Programski jezik C

3.1.1 Narišite primer diagrama poteka za while/do while/for/if stavek.

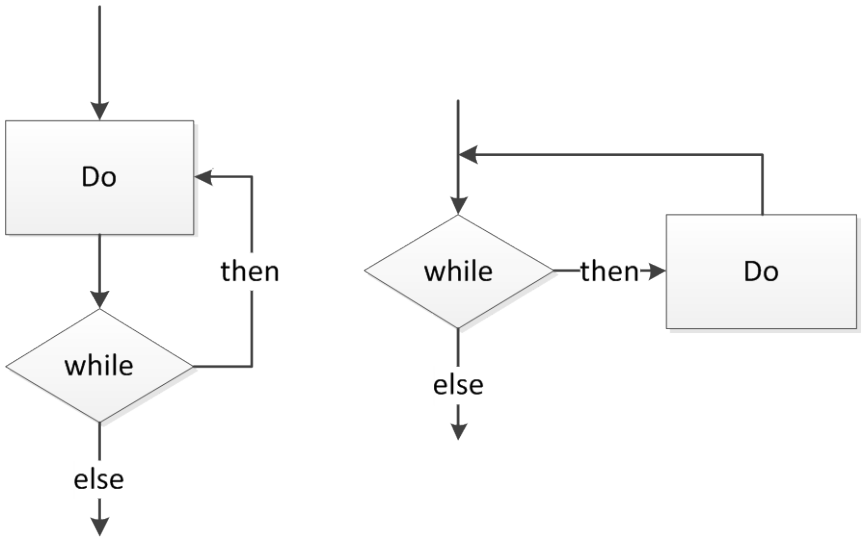
Stavek if, if_else:



Slika 1 If, If_else flowcahrt (vir: G. Nikolić, 2012)

Stavek if se povpraša po izpolnjenem pogoju, v kolikor je pogoj izpolnjen, nadaljuje svojo pot v smeri then, v kolikor pogoj ni izpolnjen nadaljuje pot v smeri else.

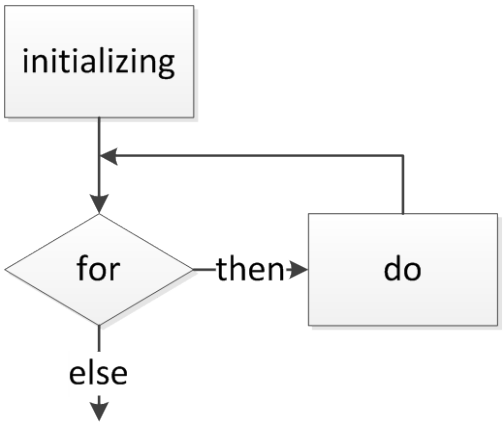
Stavek While-Do, Do-While:



Slika 2 While_Do, Do_While flowchart (vir: G. Nikolić, 2012)

While Do in Do While, sta podobni zanki, razlika je le ta, da Do While najprej izvede ter se nato povpraša po izpolnjenem pogoju, While Do pa se najprej vpraša ali je pogoj izpolnjen in šele nato izvede.

Stavek For:



Slika 3 For flowchart (vir: G. Nikolić, 2012)

For stavek ima tri vpisane argumente, najprej inicializacijo argumenta, nato pogoj za izvajanje in korak, ki se izvede ob izpolnjenem pogoju.

3.1.2 Na podlagi podanega diagrama poteka napišite program v C-ju (opomba: poudarek bo na if, while, do while in for stavku).

Zapišeš program na podlagi diagrama poteka.

3.1.3 Na kakšne načine lahko določamo tip podatkov (za znake, za cela števila in za realna števila)?

Tip podatkov določamo glede na vrsto uporabe in sicer:
- Za znake – Char (%c)
- Za cela števila – Integer (%d)
- Za realna števila – Float (%f)

3.1.4 Napišite primer določitve za konstanto (PI=3.14159)!

Določitev konstante pi katere vrednost je realno število, moramo določiti tipa float. Primer določitve:

```
const float pi=3.14159;
```

Slika 4 Primer določitve konstante (vir: G. Nikolić, 2012)

3.1.5 Kakšna je razlika med avtomatičnimi in zahtevanimi pretvorbami med tipi? Napišite primer!

Primer, ko se izvede avtomatska pretvorba med tipi:

```
int a;
float b;

a = b;
```

Slika 5 Primer avtomatske pretvorbe (vir: G. Nikolić, 2012)

V tem primeru se zgodi avtomatska pretvorba in se bo v a vpisala vrednost b-ja tipa integer. Obstajajo nekatera pravila, npr.:

a	b	a / b
int	int	int
double	int	double
int	double	double
double	double	double

Primer, ko se izvede zahtevana pretvorba med tipi;

```
int a;
float b;

b = (float)a;
```

Slika 6 Primer programa (vir: G. Nikolić, 2012)

V tem primeru smo sami zahtevali, da se v spremenljivko b zapiše vrednost a-ja kot float.

3.1.6 Kakšne so razlike med globalnimi in lokalnimi spremenljivkami? Pokažite na primeru!

Globalna spremenljivka že samo ime pove je spremenljivka, katero lahko uporabimo kjerkoli v programu. Lokalne spremenljivke, pa se uporabljajo le na posameznih odsekih. Primer definicije globalne in lokalne spremenljivke:

```
static v;

int main ( )
{
    int a=2, b=4;

    v = a + b;

    int c;

    c = v * 4;
}
```

Slika 7 Globalna in lokalna spremenljivka (vir: G. Nikolić, 2012)

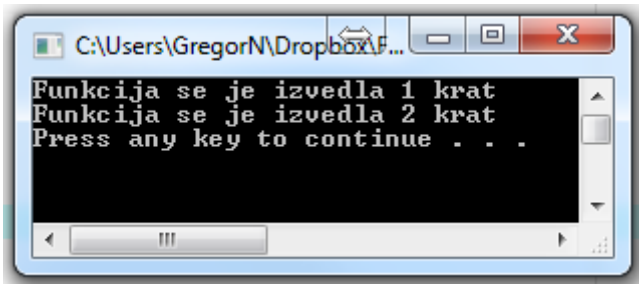
Na začetku smo definirali globalno spremenljivko v, katero lahko uporabimo kadarkoli v programu za razliko od spremenljivk a, b in c, katere pa lahko uporabimo le od mesta naprej, kjer smo jih definirali.

3.1.7 V katerem primeru spremenljivka obdrži vrednost tudi po izhodu iz funkcije in zakaj je to potrebno?

Spremenljivka katero definiramo s pomočjo ukaza S**tatic** obdrži svojo vrednost tudi po izhodu iz podprograma oz. preddefinirane funkcije. To potrebujemo kadar želimo, da je vrednost konstantna in se ne spreminja, na primer za štetje.
Primer definicije spremenljivke, ki obdrži vrednost:

```
#include <stdio.h>
#include <stdlib.h>

void stetje()
{
     static int a=0;
     printf("Funkcija se je izvedla %d krat\n",++a);
}

int main(void)
{
    stetje();
    stetje();

    system ("PAUSE");
    return 0;
}
```

Slika 8 Primer definicije spremenljivke, ki obdrži vrednost (vir: G. Nikolić, 2012)



Slika 9 Izvajanje programa (vir: G. Nikolić, 2012)

Ko se program prične izvajati kliče funkcijo stetje kjer izpiše besedilo ter poveča spremenljivko a za 1, nato se vrne iz funkcije, katero glavni program kliče še enkrat. Ker je spremenljivka a definirana kot static, se je njena vrednost po izhodu funkcije obdržala in zato je tokrat njena vrednost 2.

3.1.8 Kaj so oštevilčeni tipi? Napišite primer za "visoko, srednjo, nizko temperaturo"!

Oštevilčeni tipi ali enum. Enum je za definicijo oštevilčenih tipov in ima naslednjo obliko:

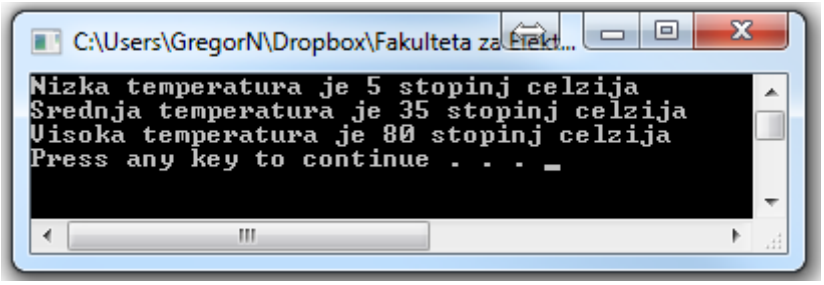
```
enum etiketa {seznam vrednosti};
enum etiketa ime_spremenljivke;
```

Slika 10 Enum definicija (vir: G. Nikolić, 2012)

Primer uporabe enum:

```
int main(void)
{
    enum temperatura {
        visoka=80, srednja=35, nizka=5 };
        /* Definirali smo seznam temperatura, katerega
        elementi so visoka, srednja in nizka ter jim kar
        priredili vrednosti.*/

    enum temperatura temp;
        /* S tem smo definirali spremenljivko temp s katero
        bomo operirali s seznamom, kateri je definiran
        zgoraj*/

    printf("Nizka temperatura je %d stopinj celzija\n", temp=nizka);
    printf("Srednja temperatura je %d stopinj celzija\n", temp=srednja);
    printf("Visoka temperatura je %d stopinj celzija\n", temp=visoka);

    system("PAUSE");
    return 0;
}
```

Slika 11 Primer uporabe enum (vir: G. Nikolić, 2012)



Slika 12 Izvajanje programa (vir: G. Nikolić, 2012)

3.1.9 Kaj so funkcije in zakaj jih uporabljamo?

Funkcije sestavljajo;

- Ime funkcije
- Tip oz. vrednost funkcije, ki jo funkcija vrne
- Vrednost, ki jo funkcija vrne
- Jedro

Funkcije so tudi preddefinirani procesi, katere lahko poljubno krat kličemo. Primer preddefiniranega procesa je lahko funkcija za seštevanje polj saj ne obstaja ukaz, ki bi seštel poljubno število polj, proces, ki izračuna površino nekega lika, proces za zakasnitev, ipd.

3.1.10 Opišite strukturo funkcije!

Strukturo funkcije bomo najlažje opisali na primeru:

```
float kvocient (float a, float b)
{
      float c;

      c = a / b;

      return c;

}
```

Slika 13 Struktura funkcije (vir: G. Nikolić, 2012)

Direktiva na začetku **float** je tip funkcije (float je decimalno število), ime **kvocient** je ime funkcije, torej če jo želimo poklicati jo pokličemo po tem imenu, v oklepaju **( )** se nahaja deklaracija vhodnih spremenljivk, v tem primeru spremenljivki **a** in **b**, katere smo definirali tipa **float**. Sledita zavita oklepaja na začetku in koncu funkcije **{ }**, označujeta začetek in konec oz. telo funkcije. V telesu se nahaja lokalna spremenljivka **c**, sledi izračun kvocienta med spremenljivko a in b ter rezultat se shrani v spremenljivko c. Kaj bo funkcija po izhodu iz funkcije vrnila je določeno z zadnjim ukazom **return c**, torej vrnila bo rezultat vrednosti spremenljivke c.

3.1.11 Opiši deklaracijo in definicijo funkcije!

Funkcija je lahko hkrati definirani in deklarirana ali pa moramo to storiti ločeno, odvisno je od načina pisanja programov.

Kadar želimo definirati in deklarirati funkcijo hkrati jo napišemo pred glavnim programom. Primer;

```
void izpis (int a, int b)
{
     printf("a = %d\nb = %d\n", a, b);
}

int main ()
{
    int g=5, n=6;
    izpis(g,n);

    system("PAUSE");
    return 0;
}
```

Slika 14 Primer sočasne definicije in deklaracije funkcije (vir: G. Nikolić, 2012)

Vidimo, da je funkcija definirana in deklarirana pred glavnim programom ter jo kličemo nemoteno. Če bi celotno funkcijo **void** prestavili za program **main** ter bi jo želeli klicati, je program nebi našel saj bi funkcija bila definirana a vendar nikjer nebi bila deklarirana. Pa si poglejmo kako to pravilno storimo:

```
void izpis (int a, int b);

int main ()
{
    int g=5, n=6;
    izpis(g,n);

    system("PAUSE");
    return 0;
}

void izpis (int a, int b)
{
     printf("a = %d\nb = %d\n", a, b);
}
```

Slika 15 Primer definicije in deklaracije funkcije (vir: G. Nikolić, 2012)

V tem primeru smo funkcijo deklarirali pred glavnim programom, da smo programu povedali, da funkcija obstaja in da jo bo ob klicu našel. Ne pozabimo, da je pri deklaraciji obvezno podpičje **;** . Funkcija katero kličemo, pa se sedaj nahaja za glavnim programom.

3.1.12 Napišite konkreten primer funkcije v programskem jeziku (void/vrne eno število/vrne več števil)!

Kot primer z uporabo funkcije, bomo zapisali program, kateri bo vsebofal funkcijo void, ki izpiše eno ter void, ki izpiše več števil. Uporabili bomo deklaracijo in definicijo funkcije posebej.

```
void eno_st(int a);                          // Deklaracija funkcije
void vec_st(int a, int b, int c, int d);     // Deklaracija funkcije

int main ()                                  // Glavni program
{
    printf("Izpis ene vrednosti: ");         // Izpis stavka
    eno_st(500);                             // Vnos vrednosti in izpis slednje
    printf("\n\n");

    printf("Izpis vecih vrednosti: ");       //Izpis stavka
    vec_st(100, 150, 200, 250);              // Vnos vec vrednosti in izpis slednjih
    printf("\n\n");

    system("PAUSE");
    return 0;


}

void eno_st(int a)                           // Definicija funkcije, ki vraca
{                                            // eno v rednost
     printf("%d",a);
}

void vec_st(int a, int b, int c, int d)      // Definicija funkcije, ki vraca
{                                            // vec vrednosti
     printf("%d %d %d %d", a, b, c, d);
}
```

Slika 16 Primer programa, ki vrača vrednost (vir: G. Nikolić, 2012)

3.1.13 Kakšne so prednosti in slabosti uporabe funkcij?

Ena izmed prednosti uporabe funkcij je zagotovo preglednost programa, omogoča večkratno uporabo dela kode, s tem pridobimo na dolžini programa ter na času pisanja programa toda slabost tega je počasnejše izvajanje, trajanje skokov v funkcije, ipd.

3.2 Vhodi/Izhodi funkcije

3.2.1 Možno vprašanje A: Koliko vhodov in izhodov imajo funkcije?

Funkcije imajo vhodov toliko, kolikor jih definiramo, izhod je pa le eden. Funkcija lahko vrne le eno vrednost.

3.2.2 Možno vprašanje B: Imajo funkcije vhodne/izhodne spremenljivke in kako jih realiziramo?

Funkcije imajo lahko več vhodnih spremenljivk, izhodna je le ena. Vhodne spremenljivke definiramo v »glavi« funkcije, med oklepajema, izhodno funkcijo pa definiramo v telesu funkcije katero tudi vrnemo z ukazom return ime_funkcije.

3.2.3 Na katere tri načine lahko uporabimo funkcije v programu?

Funkcije v programu lahko uporabimo tako, da:

- Jih definiramo in deklariramo hkrati (zapis funkcije pred glavnim programom),
- Jih najprej pred glavnim programom deklariramo in nato za glavnim programom definiramo,
- So že preddefinirane, pomeni, da so že napisane (deklaracije se nahajajo v zaglavjih, ena pogostih preddefiniranih funkcij v C-ju je printf, ki se nahaja v zaglavju stdio.h).

3.2.4 Kaj pomeni izraz "zaglavje" v C programu in kako jih uporabljamo?

Zaglavje so v naprej napisani programi, ki se ne nahajajo v samem programu, temveč jih vključujemo v program. Se nahajajo v drugi datoteki. Primer:

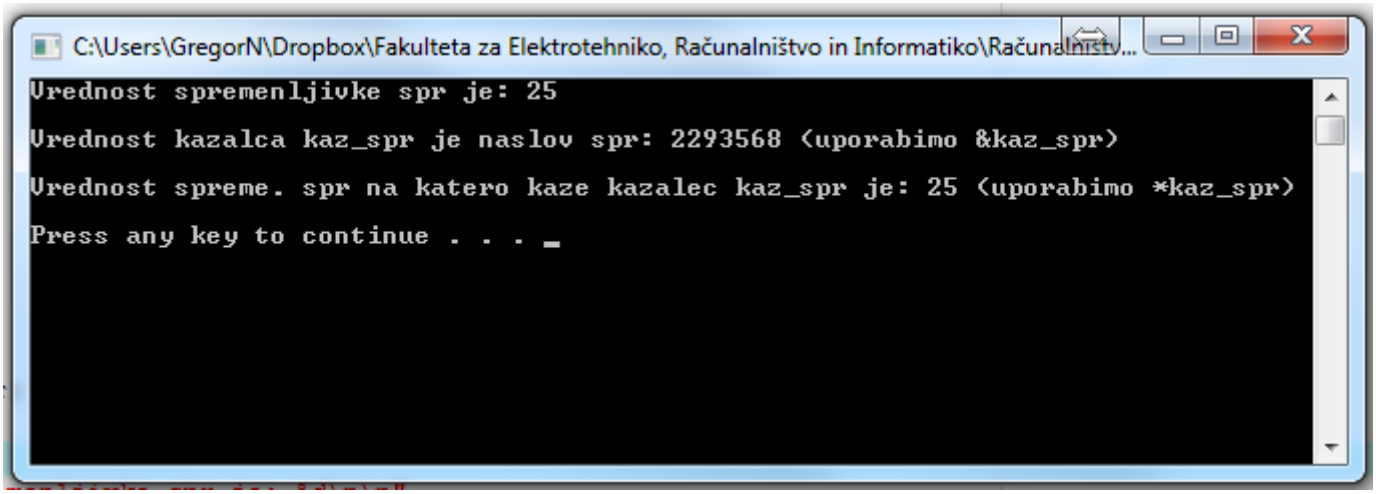
Če želimo izpisati nek stavek, uporabimo funkcijo printf, ta funkcija se nahaja v datoteki stdio.h katero vključimo v program na način #include <stdio.h>.

3.2.5 Kaj so kazalci in kako jih uporabljamo ter predstavljamo? Kaj so vrednosti, naslov in vsebina kazalca? Pokažite na primeru!

Kazalci so spremenljivke, katere vsebujejo naslov neke druge spremenljivke, katera pa vsebuje vrednost. Torej kazalci le kažejo na druge spremenljivke katere vsebujejo vrednost.

```
int main()
{
    int spr;         // Spremenljivka tipa integer
    int *kaz_spr;    // Kazalec, ki kaze na integer

    spr = 25;        // Dodelitev vrednosti spr

    kaz_spr = &spr; // Kazalec dobi naslov od spr

    printf(
    "Vrednost spremenljivke spr je: %d\n\n"
    "Vrednost kazalca kaz_spr je naslov spr: %d (uporabimo &kaz_spr)\n\n"
    "Vrednost spreme. spr na katero kaze kazalec kaz_spr je: %d (uporabimo *kaz_spr)\n\n"
    , spr, &kaz_spr, *kaz_spr);

    /*
    Za izpis vrednosti kazalca uporabimo znak &, za izpis vrednosti
    spremenljivke na katero kaze kazalec pa uporabimo znak *.
    */

    system("PAUSE");
    return 0;
}
```

Slika 17 Primer programa z uporabo kazalcev (vir: G. Nikolić, 2012)



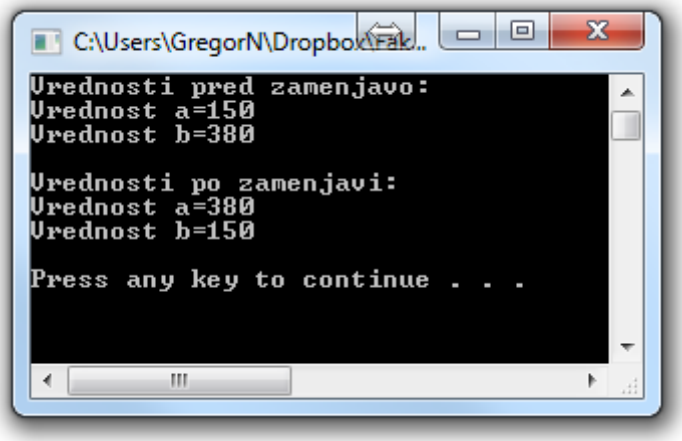
Slika 18 Izvajanje programa (vir: G. Nikolić, 2012)

3.2.6 Kako uporabimo kazalce, da dobimo izhodne spremenljivke funkcij?

Da želimo na primer zamenjati dve vrednosti spremenljivk med seboj, bi uporabili funkcijo katera bi zamenjala števili ter nam dala rezultat le ene, v ta namen lahko uporabimo kazalce spremenljivk in zamenjamo vrednosti s pomočjo kazalcev in funkcije, ki ne vrača vrednosti. Primer:

```
void zam( int *kaz_a, int *kaz_b )
{
     int temp;

     temp = *kaz_a;
     *kaz_a = *kaz_b;
     *kaz_b = temp;
}

int main ()
{
    int a=150, b=380;

    printf(
    "Vrednosti pred zamenjavo:\n"
    "Vrednost a=%d\n"
    "Vrednost b=%d\n", a, b);

    zam(&a,&b);

    printf(
    "\nVrednosti po zamenjavi:\n"
    "Vrednost a=%d\n"
    "Vrednost b=%d\n\n", a, b);

    system("PAUSE");
    return 0;
}
```

Slika 19 Primer programa z uporabo kazalcev za vračanje več vrednosti iz funkcij (vir: G. Nikolić, 2012)



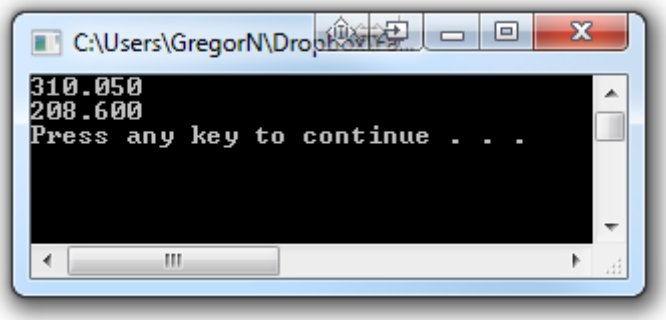
Slika 20 Izvajanje programa (vir: G. Nikolić, 2012)

3.2.7 Ali lahko uporabimo kazalce na funkcije? Navedite primer!

Kazalce, ki kažejo na funkcije lahko uporabimo. Primer uporabe kazalca, ki kaže na funkcijo, zraven bomo napisali še klic funkcije brez kazalca za primerjavo.

```
float (*operacija)(float,float); // Deklaracija kazalca *operacija, ki kaze na
                                 // funkcijo, ki ima dva vhodna podatka tipa
                                 // float

float vsota (float a, float b);  // Deklaracija funkcije

int main()                       // Glavni program
{                                // Zacetek glavnega programa
    float x=104.5, y=205.55, c;  // Definicija in deklaracija spremenljivk

    c = vsota(x,y);              // Klic funkcije vsota in zapis vrnjene
                                 // vrednosti v spremenljivko c
    printf("%.3f \n", c);        // Izpis vrednosti spremenljivke c

    operacija = &vsota;          // Zapis NASLOVA funkcije v kazalec OPERACIJA
    c = operacija(54.4, 154.2);  // Klic funkcije VSOTA preko kazalca, ki
                                 // kaze na funkcijo VSOTA
    printf("%.3f \n", c);        // Izpis vrednosti spremenljivke c

    system("PAUSE");
    return 0;
}


float vsota (float a, float b)   // Definicija funkcije, ki izracuna vsoto
{
    return (a+b);
}
```

Slika 21 Primer programa z uporabo kazalcev na funkcije (vir: G. Nikolić, 2012)



Slika 22 Izvajanje programa (vir: G. Nikolić, 2012)

3.2.8 Opišite vektorje in polja v programskem jeziku C!

Vektorji so enodimenzionalne matrike. Primer enodimenzionalne matrike oz. vektorja:

$$v = \begin{bmatrix} 3, & 5, & 8, & 3, & 1 \end{bmatrix}$$

Polja so zgolj sestavljena iz več vektorjev. Vsi elementi vektorjev in polj so istega tipa. Primer polja:

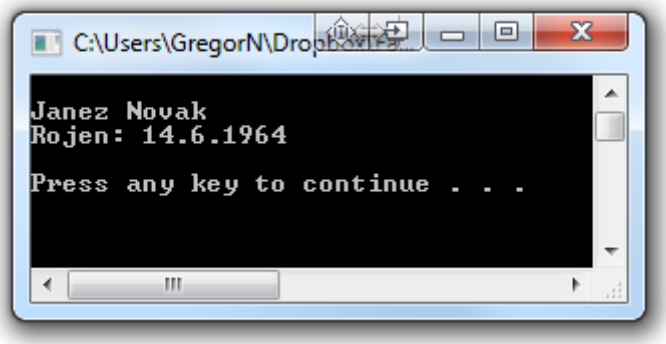
$$M = \begin{bmatrix} 5 & -4 & 3 \\ 2 & 16 & 21 \\ 2 & 0 & 1 \end{bmatrix}$$

3.2.9 Opišite strukture v programskem jeziku C!

Strukture so zelo podobne vektorjem, vendar strukture sestavljajo različni tipi. Strukture definiramo tako, da ji damo ime in naštejemo vse elemente kateri jo sestavljajo. Primer uporabe strukture:

```
struct Osebni_podatki        // Definicija strukture
{
       char ime[20];         // Deklaracija in definicija niza 20ih znakov
       char priimek[20];     // Deklaracija in definicija niza 20ih znakov
       int dan_rojstva;      // |
       int mesec_rojstva;    // |
       int leto_rojstva;     // \->
}/*struct Osebni_podatki joze; <- zdruzili bi tukaj*/;

int main()
{
    struct Osebni_podatki janez;       // Deklaracijo lahko zdruzimo ze v deviniciji
    /*     Ime strukture  Ime deklarirane spremenljivke*/
    strcpy(janez.ime,"Janez");          // Vnos "Joze" v niz ime
    strcpy(janez.priimek,"Novak");// Vnos "Marjetica" v niz priimek
    janez.dan_rojstva=14;              // Vnos vrednosti v struk. dan_rojstva
    janez.mesec_rojstva=06;             // Vnos vrednosti v struk. mesec_rojstva
    janez.leto_rojstva=1964;           // Vnos vrednosti v struk. leto_rojstva

    printf("\n%s %s\n",janez.ime,janez.priimek);// Izpis imena in priimka
    printf(
    "Rojen: %d.%d.%d\n\n"
    ,janez.dan_rojstva,janez.mesec_rojstva, janez.leto_rojstva);// Izpis podatkov rojstva

    system("PAUSE");
    return 0;
}
```

Slika 23 Primer programa z uporabo struktur (vir: G. Nikolić, 2012)



Slika 24 Izvajanje programa (vir: G. Nikolić, 2012)

3.2.10 Kako definiramo polje znakov v programskem jeziku C? Koliko znakov je v resnici v polju znakov in opišite zakaj je razlika?

Polje znakov definiramo s sintakso **char** pri kateri pa imamo dva načina definiranja polja znakov. Prvega načina se nekoliko izogibamo, saj nam je bolj praktičen drug način, kako bomo spoznali na primeru. Pomembno omeniti je, da se vsako polje znakov konča z ničlo (0). Pri drugem načinu nam te ničle ni potrebno pisati, saj jo prevajalnik doda avtomatsko. V polju znakov je število znakov toliko kolikor jih napišemo +1, ne pozabimo zadnji je ničla. Pa si poglejmo primer:

```
int main ()
{
    char niz1[] = {'G','r','e','g','o','r',' ','N','i','k','o','l','i','c','\0'};
    char niz2[] = "Gregor Nikolic";

    printf("%s\n\n", niz1);
    printf("%s\n\n", niz2);

    system("PAUSE");
    return 0;

}
```

Slika 25 Primer definiranja polja znakov (vir: G. Nikolić, 2012)

3.2.11 Navedite vsaj štiri najbolj pogosto uporabljene specifikacije (za %) za formatirano branje in pisanje!

Za branje in pisanje se uporabljata naslednja ukaza:
Za branje:

- `scanf(format, seznam naslovov spremenljivk);`
  - Primer branja: `scanf("%d", &st);`
    & znak pred spr. st se mora nahajati, saj moramo imeti podan naslov spremenljivke, kamor se bo vpisala vpisana vrednost.

- `printf(format, seznam spremenljivk ali konstant);`
  - Primer izpisa niza in vrednosti spr.: `printf("Starost: %d", st);`
    Za izpis pa za razliko od branja pred spremenljivko st ne potrebujemo znaka &, saj želimo izpisati vrednost in ne naslova spremenljivke.

Najbolj pogosti tipi, ki se uporabljajo za specifikacije so:

- %d – decimal (celo število),
- %c – char (znak),
- %s – string (niz),
- %f – float (decimalno število),
- itn.

3.2.12 Ali lahko dinamično dodeljujemo pomnilnik in kako?

Pomnilnik lahko dodeljujemo tudi dinamično in sicer s knjižnično funkcijo malloc( ). Malloc funkciji podamo kot parameter število bajtov pomnilnika, ki ga potrebujemo. Funkcija nam bo rezervirala želeno količino pomnilnika ter vrnila naslov začetka rezerviranega pomnilniškega bloka. V koliko v sistemu ni dovolj pomnilnika, funkcija vrne vrednost NULL.

3.2.13 Bitne operacije - kako je izvedeno branje enega bita?

Za branje enega bita uporabimo maskiranje ter bitni and (IN) &.

	1	0	0	1	1	0	1	1	Spremenljivka
&	0	0	0	0	1	0	0	0	Maska
	0	0	0	0	1	0	0	0	Rezultat

Slika 26 Branje enega bita (vir: G. Nikolić, 2012)

V C programskem jeziku bi zapisali:

```
unsigned int rezultat;
unsigned int spremenljivka = 0x9B;

rezultat = spremenljivka & 0x08;
```

Slika 27 Primer C programa za branje bita (vir: G. Nikolić, 2012)

3.2.14 Bitne operacije - kako je izvedeno brisanje enega bita?

Prav tako kot za branje enega bita uporabimo za brisanje enega bita tudi maskiranje ter bitni and (IN) &, le da tokrat uporabimo drugačno vrednost.

	1	0	0	1	1	0	1	1	Spremenljivka
&	1	1	1	1	0	1	1	1	Maska
	1	0	0	1	0	0	1	1	Rezultat

Slika 28 Brisanje enega bita (vir: G. Nikolić, 2012)

V C programskem jeziku bi zapisali:

```
unsigned int rezultat;
unsigned int spremenljivka = 0x9B;

rezultat = spremenljivka & 0xF7;
```

Slika 29 Primer C programa za brisanje enega bita (vir: G. Nikolić, 2012)

3.2.15 Bitne operacije - kako je izvedeno postavljanje enega bita?

Za postavljanje enega bita uporabimo bitno funkcijo or (ALI) |.

	1	0	0	1	1	0	1	1	Spremenljivka
|	0	0	1	0	0	0	0	0	Maska
	1	0	1	1	1	0	1	1	Rezultat

Slika 30 Postavljanje enega bita (vir: G. Nikolić, 2012)

V C programskem jeziku bi zapisali:

```
unsigned int rezultat;
unsigned int spremenljivka = 0x9B;

rezultat = spremenljivka | 0x20;
```

Slika 31 Primer C programa za postavljanje enega bita (vir: G. Nikolić, 2012)

3.3 Postopki pri prevajanju programov

3.3.1 Kaj vse potrebujemo za izvedbo postopka prevajanja?

Za postopek prevajanja potrebujemo:

- Prevajalnik (Assembler),
- povezovalnik (Linker),
- knjižnice (Library) in
- generator binarne kode (Executable file).

3.3.2 Kaj so prevajalnik, povezovalnik, knjižnice in generator binarne kode?

Prevajalnik nam preslika program v nov program v strojnem jeziku. Vredno je omeniti, da je strojni jezik najnižji programski jezik.
Povezovalnik poveže objektno kodo s knjižnico, izdela izvajalsko datoteko ter jo shrani na disk.
Knjižnice vsebujejo skupek funkcij, konstant, objektov, itd. katere kličemo z ukazi. Knjižnice se nahajajo v zaglavju programa.
Generator binarne kode nam izdela izvršilno datoteko.

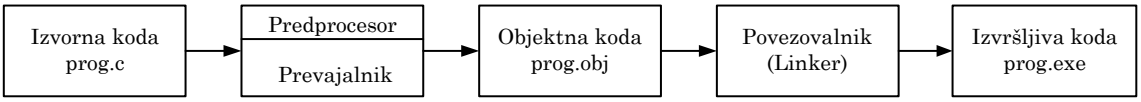
3.3.3 Kaj so mnemoniki pri asemblerju in zakaj jih uporabljamo?

Mnemonik koda ali simboli pri Assemblerju so pravzaprav ukazi, po navadi od 1 do 5 črk katere predstavljajo opcode (izvršilna koda). Primer:

movlw – move literar to work – premakni vrednost v delovni register Work.

Mnemonike uporabljamo enako kot programski jezik C za programiranje in pisanje programa. Razlika med njima je, da je Assebler programski jezik, najnižji programski jezik, takoj pred strojnim jezikom.

3.3.4 Opišite potek prevajanja!

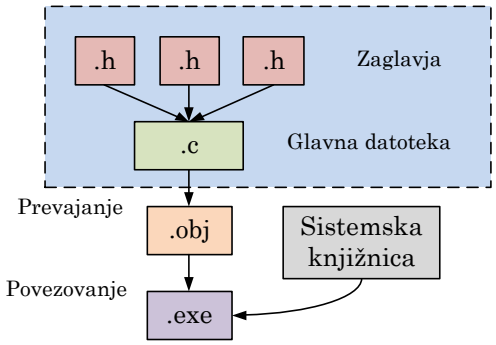


Slika 32 Blokovna shema prevajanja programa (vir: G. Nikolić, 2012)

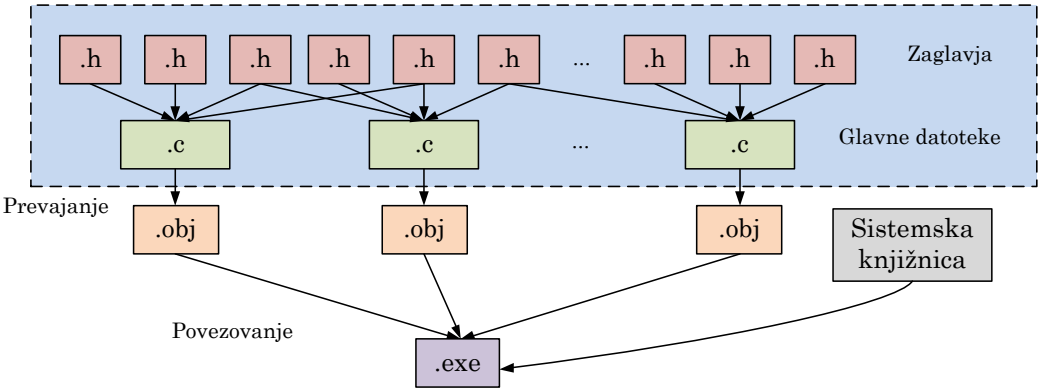
Najprej zapišemo program v programskem jeziku, naj bo to programski jezik C. S prevajalnikom prevedemo napisan program v objektno kodo katero s pomočjo povezovalnika (Linker) izdelamo izvršljivo kodo.

3.3.5 Kako poteka prevajanje enostavnega in bolj zapletenega programa?

Prevajanje enostavnega programa poteka tako, da iz več zaglavij ter ene glavne datoteke poteka prevajanje v objektno kodo, ter nato preko povezovalnika v izvršljivo kodo.
Pri zapletenejšem programu se prevajanje vrši preko več zaglavij, katera se lahko medsebojno povezujejo z več glavnimi datotekami, tako prevajalnik prevede več glavnih datotek v več objektnih kod, katere se nato s pomočjo povezovalnika združijo v izvršilno datoteko.



Slika 33 Blokovna shema prevajanja enostavnega programa (vir: G. Nikolić, 2012)



Slika 34 Blokovna shema prevajanja zahtevnega programa (vir: G. Nikolić, 2012)

3.3.6 Kaj je razhroščevalnik (debugger) in kako deluje?

Razhroščevalnik ali ang. debugger je program, s katerim preizkušamo in razhroščujemo druge programe. Razhroščevalnik nam pokaže napake v programski kodi. Omogoča nam tudi tipično izvajanje korak po koraku ter uporabo zaustavitve izvajanja s break pointi (breakpoint).
Morda si bistvo debuggerja oz. razhroščevalnika lažje zapomnimo, če vemo kako je to ime nastalo; Leta 1945 na Harvard University se je pokvarilo računalo Mark II Aiken Relay Calculator in sicer zaradi napake, ker se je med kontakte releja ujel ščurek ang. bug.

3.3.7 Kako najdemo napake v programu s pomočjo razhroščevalnika (debuggerja)?

Razhroščevalnik nam pomaga iskati logične napake katere smo naredili ob pisanju programa. Razhroščevalnik nam ne pomaga pri tem, da bo program v resnici deloval kot smo si zamislili, pomaga nam s koraki preverjanja vrstice za vrstico, ali pa prevedemo odsek katerega določimo s t.i. break pointi.
Po uspešnem pregledu in odpravi napak z razhroščevalnikom ni nujno, da bo program deloval po zastavljenih ciljih saj razhroščevalnik ne rešuje smiselnosti in delovanja programa temveč išče napake v programu.

3.3.8 Kaj je simulator in kako deluje? Navedite primer!

Zamislimo si, da smo napisali program, ki nam vsaki dan ob 14h odpre okno za 2h. Za preizkus programa bi morali sestaviti celoten mehanizem, integrirano vezje ipd. v katero bi vpisali program, nato pa še čakali na ustrezno uro, da bi videli ali program deluje. Da nam ni potrebno tega storiti le zato, da bi na koncu ugotovili, da smo kje storili napako, v ta namen uporabimo t.i. simulatorje.
Simulator je program v katerem simuliramo sestavne dele razvojnem okolju v katerih se odvija naš program. Vsi sestavni deli so simulirani, torej neresnični. Tudi simulator nam včasih ne daje popolnih podatkov o tem ali bo zadeva v ciljnem sistemu delovala, zato je dobro narediti prototip.
Sam kot raziskovalec uporabljam za uspešno izdelane izdelke postopek: Pisanje programa, razhroščevanje, simuliranje v simulatorju, sestava vezja oz. sistema kot prototip in šele nato po uspešnem testiranju končni izdelek.

3.4 Pristopi h kvalitetnemu programiranju

3.4.1 Kakšne značilnosti ima kvalitetna programska oprema?

Da je programska oprema kvalitetna mora biti preprosta za:

- Razumevanje avtorja in ostalih
- odpravljanje napak,
- verifikacijo (dokazovanje pravilnosti delovanja) in
- vzdrževanje (dodajanje funkcij in spreminjanje).

3.4.2 Katero je zlato pravilo razvoja programske kode? Utemelji odgovor!

Zlato pravilo razvoja programske kode je, da naj programer piše program tako kot bi si želel, da bi nekdo napisal program zanj. Zakaj je to pomembno? Če napišemo program, ki deluje popolnoma pravilno iz zelo zadovoljivo, je problem, če ga razumemo le mi, zato je pomembno, da je napisan razumljivo tudi za tiste, ki niso vešči programiranja.

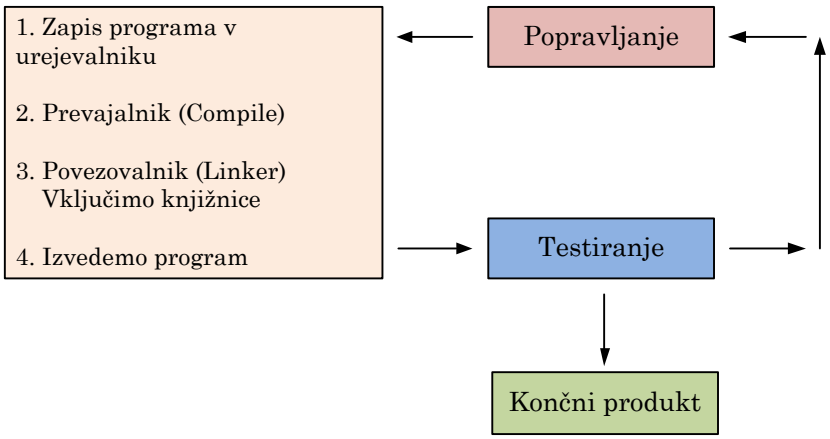
3.4.3 Katere so pet zapovedi za uspeh pri delu?

Pet zapovedi, ki jih je smiselno upoštevati so:

1. Ne začnimo s preobsežnimi načrti.
2. Preden se lotimo dela, dobro premislimo o njem.
3. Ne hitimo pri delu (komur se mudi je na najboljši poti, da bo naredil napako).
4. Ne bodimo skopi kadar kupujemo orodja (pogledamo reference, teste,..). Slaba orodja (nekvalitetna) nam onemogočajo hitro in učinkovito delo.
5. Ne lotevajmo se vedno vsega sami!

3.4.4 Katere glavne faze razvoja programa poznamo?

Razvoj opreme je koračen postopek katerega najlažje opišemo s sliko:



Slika 35 Blokovna shema faz razvoja (vir: G. Nikolić, 2012)

3.4.5 Kaj in koga analiziramo na začetku razvoja programa?

Na začetku razvoja programa moramo najprej analizirati nalogo katere se lotevamo. Šele po dobri analizi nadaljujemo na razvoju programa.

3.4.6 Opišite kako in zakaj uporabljamo ocenjevanje?

Ocenjevanje delovanja programa je pomembno, saj je od njega odvisno delovanje končnega programa. Ocenjevanje programa je najbolje dati osebi, ki ni nikoli programirala, da ga preizkusi, saj bo tako najhitreje prišlo do situacije, ko bo razvidna kakšna napaka, saj oseba razmišlja drugače kot programer.

3.4.7 Kako deluje top-down/(bottom up) metoda?

Top down in bottom up ali od zgoraj navzdol oz. od spodaj navzgor metoda načrtovanja.

- Načrtovanje od zgoraj navzdol (top-down) je metoda, kjer najprej definiramo celotno nalogo, nato pa nadaljujemo z definiranjem pod-nalog.
- Načrtovanje od spodaj navzgor (bottom-up) pa je metoda, kjer najprej začnemo s podrobnostmi in nato pod-naloge povezujemo, da dobimo rešitev celotne naloge.
-

3.4.8 Kako lahko poimenujemo besede pri programiranju in na kaj moramo paziti pri poimenovanju?

Spremenljivke oz. besede poimenujemo smiselno za kar so uporabljene. Izogibamo se velikim in malim črkam oz. poimenovanju besed katerih so razlike le velike oz. male črke. Najbolje je pisati le z malimi črkami in se tako izognemo večini napak, saj v večini primerov pozabimo kje smo pisali z veliko ter kje ne.

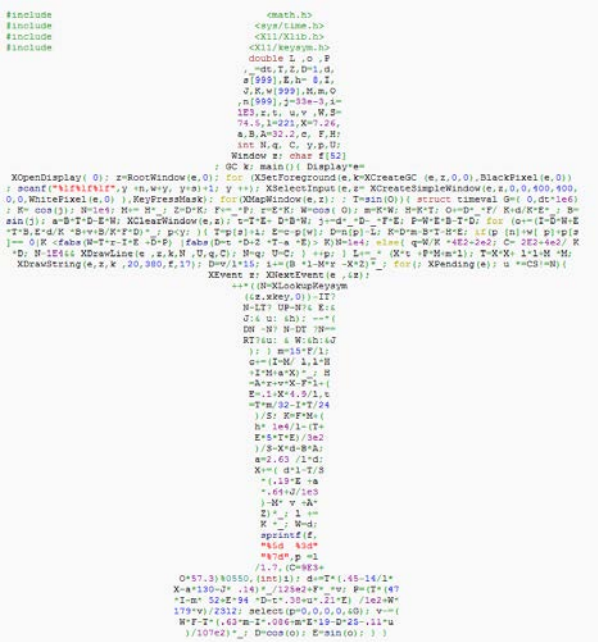
3.4.9 Kaj se zgodi, če delimo z 0? Utemeljite odgovor?

Deljenje z ničlo je že v matematiki nedefiniran izraz, enako velja za programski jezik in program. V primeru, da se zgodi delitev z ničlo pride navadno do zrušitve programa.

3.4.10 Kako naredimo program "berljiv"?

Z besedo berljiv imamo v mislih preglednost programa. Čeprav obstajajo tekmovanja za najbolj neberljivo napisan delujoč program, se poslužujemo lepo preglednega pisanja programa. Kakšen je lepo napisan program? Držimo se pravila, da pišemo telesa zank, ukaze ipd. za vsaj en tabulatorski presledek v desno od levega roba. Najbolje bo to razvidno s primerom:

Primer nepregledne kode:



Slika 36 Zmagovalec tekmovanja International Obfuscated C Code Contest leta 1998 (vir: Wikipedia)

Primer pregledno napisanega programa:

```c
/*
Zapiši program, ki bo iz poljubnega decimalnega stevila, izpisal prva tri stevila za vejico.
Primer:
Poljubno decimalno stevilo 124,56786 => rezultat 567
*/

#include <stdio.h>          //Vkljucitvene datoteke
#include <stdlib.h>         //Vkljucitvene datoteke

int main()                  //Glavni program
{                           //Zacetek glavnega programa
    double ost, st;         //Deklaracija spremenljivk ost in st kot double zaradi velikosti stevila
    int cst;                //Deklaracija spremenljivke cst kot integer - celo stevilo

    printf("Prosim vpisite decimalno stevilo. Pri vpisu uprabite piko.\n\n\t");       //Izpis stavka
    scanf("%lf", &st);      //Vpis decimalnega stevila
    printf("\nVpisali ste stevilo: %lf\n\n",st);      //Izpis stavka in vpisanega stevila
    cst=st;                 //Premakne decimalno stevilo v spremenljivko st - tako se resimo decimalnih mest
    ost=st-cst;             //Odstejemo celo stevilo od decimalnega stevila in ostanejo le decimalke
    ost=ost*1000;           //Premaknemo cedimalke za tri mesta v levo - da pridbimo prva tri stevila
    printf("Prva tri stevila za decimalno vejico so: %d\n\n", (int)ost);  //Izpisemo stavek ter prva tri
                                                                          //decimalna stevila kot tip integer,
                                                                          //da se izognemo zaokrozevanju

    system("PAUSE"); //Zaustavitev konzole
    return 0;        //Uspesno preveden program
}                    //Kondec glavnega programa
```

Slika 37 Primer pregledno napisanega programa (vir: G. Nikolić, 2012)

3.4.11 Katere so najbolj pogoste napake pri pisanju programov?

Najbolj pogoste napake pri pisanju programov so:

- Zaradi praznih mest v operatorju
  - „! =“ namesto „!=“
- Zaradi nepravilnega vrstnega reda
  - „=>“ namesto „>=“
  - „=!“ namesto „!=“
- Nepravilna uporaba „=“ in „==“.
- Pri uporabi operatorjev uporabimo raje oklepaje, namesto, da ugibamo prioriteto slednjih.

3.4.12 Naštejte in natančno opišite vsaj štiri pravila kvalitetnega programiranja!

1. Besede za spremenljivke morajo biti izbrane tako, da ni potrebnega dodatnega komentiranja programa. Primer: zn_tmp – nujno potreben komentar, ustrezneje bi bilo zunanja_temperatura.
2. Pišimo pregledno, pomeni da uporabljajmo presledke, tabulatorje, da je program pregleden.
3. Vedno pazimo, da določimo pravilen tip funkcije; int stetje (int a);
4. Inicializiramo in definiramo spremenljivke hkrati, saj s tem skrajšujemo čas izvajanja programa; int a=4;
5. Pišimo program na enostaven način, naj bo čim bolj enostaven.
6. Pri pisanju pazimo, da ne pride v programu do situacije deljenja z ničlo, saj vodi do zrušitve programa.
7. Vstavljamo pravilna zaglavja, ko delamo s funkcijami. Primer zaglavja math: #include <math.h>
8. Kazalce vedno pravilno inicializiramo in poimenujemo, da ne pride do napak. Primer: int stevec, *kaz_stevec;
9. Konstante definirajmo z ukazom const.

3.4.13 Ali je pravilno glede na zahteve po kvalitetnem programiranju (opomba: podano na izpitu/testu)?

Na vprašanje lahko odgovorimo, če poznamo primere kvalitetnega programiranja. Ko predse prejmemo program pogledamo zgradbo, preglednost, smiselnost imena spremenljivk, itd.

3.4.14 Kakšna je razlika med strukturnim in objektnim programiranjem?

Razlika med slednjim je:

- Strukturno programiranje:
  - Razčlenjujemo algoritem (jasna notranja logična struktura),
  - velike rutine razdelimo na več manjših,
  - modularno in
  - brez »go to« stavkov.

- Objektno (predmetno) programiranje:
  - Objekti, lastnosti, metode,
  - razredi objektov,
  - hierarhija in
  - dedovanje.

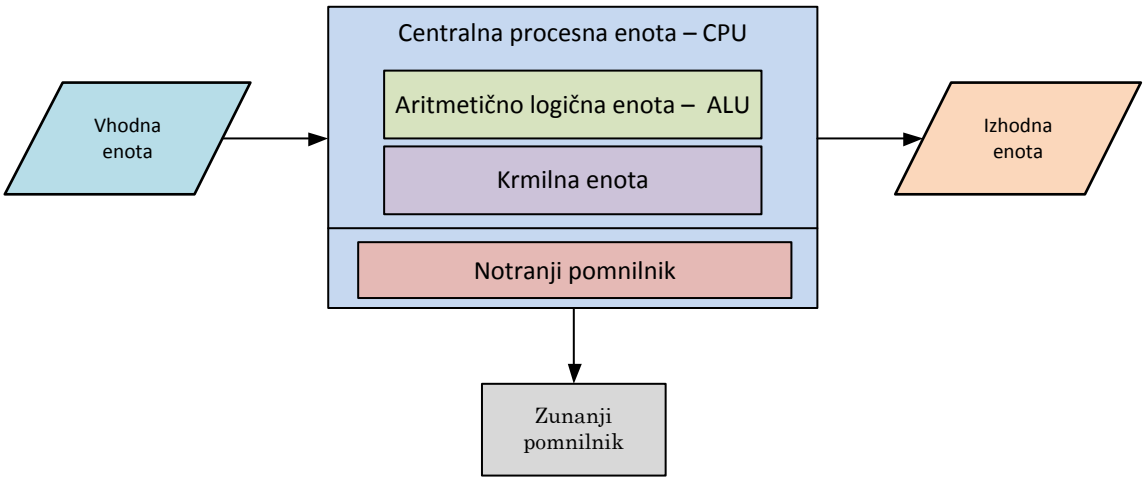
3.5 Računalniški sistem

3.5.1 Opišite osnovno zgradbo računalniškega sistema

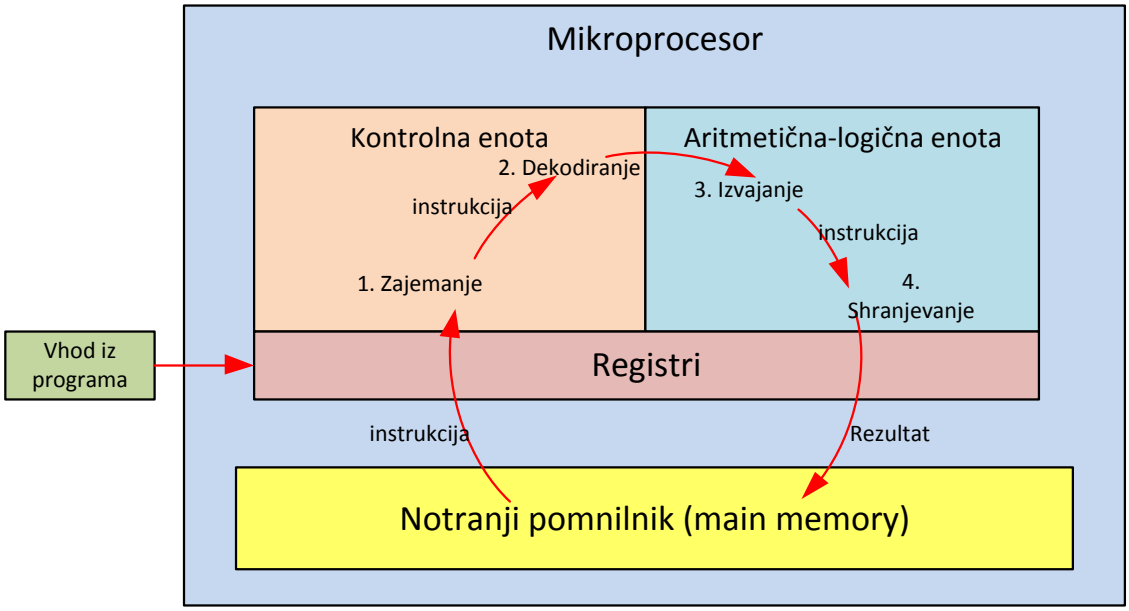
Osnovna zgradba računalniškega sistema:

- Vhodne enote (tipkovnica, miška, …),
- procesor – CPU (Centralno Procesna Enota ang. Central Processing Unit),
- razne kartice:
  - Zvočna
  - Video
  - Mrežna
- napajalnik,
- izhodne enote,
- notranji pomnilnik in
- zunanji pomnilnik.

3.5.2 Natančno opišite in narišite osnovno zgradbo računalnika in nato opišite, kako deluje centralna procesna enota (CPU)!



Slika 38 Osnovna zgradba računalnika (vir: G. Nikolić, 2012)



Slika 39 Zgradba CPU (vir: G. Nikolić, 2012)

Takoj, ko na primer dvokliknemo ikono na namizju, se sproži niz ukazov katere CPU (centralno procesna enota ali ang. Central Processing Unit) obdela tako, da iz pomnilnika pridobi podatke o nizu ukazov (1.), jih dekodira (2.) ter pošlje aritmetično-logični enoti (ALE ali ang. ALU, Arithmetic Logical Unit) katera izvede aritmetično ali logično operacijo (3.) ter nato shrani rezultat (4.) v pomnilnik.

3.5.3 Kaj je cikel procesiranja? Kako merimo njegovo hitrost in kateri so faktorji, ki vplivajo na hitrost?

En strojni cikel procesiranja so štiri operacije in sicer:

1. Zajemanje
2. Dekodiranje
3. Izvajanje
4. Shranjevanje

Izvajanje operacij se vrši po korakih, kateri pa se vršijo po časovnem taktu oz. po računalniškem taktu. Takt ali urin cikel se določi glede na hitrost delovanja računalniškega procesorja, katerega merimo v MHz oz. GHz. Samo kot primer omenimo, da mikrokrmilniku PIC ter ostalim, lahko določamo takt izvajanja z zunanjim kristalom, tudi tam se izvajajo operacije. Če imamo na mikrokrmilniku pic 4 MHz kristal in vemo, da slednji deluje z ¼ te frekvence, je čas ene operacije natanko 1 µs.
Hitrost procesorja lahko merimo tudi v t.i. MIPS-ih (Million Instructions Per Second), kar pomeni, koliko milijonov instrukcij je sposoben narediti procesor v eni sekundi.

3.5.4 Kaj sta aritmetično logična enota in krmilna enota ter kako delujeta?

V aritmetično logični enoti ali ang. Arithmetic Logichal Unit (ALU) se izvajajo, kot že ime pove aritmetične in logične operacije. Povezuje tri pomnilne celice oziroma registre in sicer za shranjevanje operandov in rezultatov ter akumolator. Izvaja le elementarne operacije, npr. seštevanje, vse ostale operacije se razbijejo ne elementarne npr. 2x8 = 8+8.

Krmilna enota je del računalnika, ki nadzoruje in usklajuje delovanje posameznih enot ter skrbi za pravilno delovanja programov. Program se izvaja koračno torej korak za korakom ter se odloča in skrbi za podajanje podatkov v ALU oz. v ustrezno enoto. Krmilna enota ugotovi pomen ukaza, ki se prenese v ukazni register, ki se nato izvede.

3.5.5 Kaj je register in kako deluje?

Register je začasni pomnilnik, kamor se shranjujejo podatki in instrukcije. Register shranjuje;

- lokacije, od koder so bile dobljene instrukcije
- instrukcije, ko so bile dekodirane,
- podatke, katere med tem ALU obdeluje in
- rezultate izračuna.

3.5.6 Kaj so dual-core in multi-core procesorji?

Dual-core v slovenščini pomeni dvo-jedrno ter multi-core, več-jedrno. So čipi oz. integrirana vezja, ki vsebujejo dual-dva jedra ali multi-več jeder. So zmogljivejši procesorji, saj si delijo delo in tako opravijo enako delo v občutno krajšem času.

3.5.7 Kaj so hladilniki?

Hladilniki so, običajno iz aluminija, kosi materiala z režami s katerimi odvajamo toploto od elektronskih komponent. Hladilnik se preko temperaturno prevodne paste pritrdi na recimo procesor. Hladilniki imajo po navadi večji koeficient temperaturne prevodnosti in tako prevzemajo temperaturo elementa katerega hladijo. Na hladilnike nato za večji efekt montiramo ventilatorje ali celo (po navadi) bakrene cevi, skozi katere teče hladna voda in tako še občutneje hladi napravo.

3.5.8 Kaj je bit, nibble, byte?

Bit je enota za merjenje množine informacij (Binary digit). Bit lahko zavzame le dve vrednosti in sicer 0 ter 1.
Skupino štirih bitov imenujemo nibble, osmih pa byte. Prenos podatkov merimo v bitih na sekundo.
Tukaj je vredno omeniti primer za boljše razumevanje; hitrost prenosa, ki ga ponujajo ponudniki interneta, imajo zapisano npr.: 10 Mb/s kar je enako 10 Mega bitov / sekundo, nato pa želimo prenesti dokument, ki je velik 100 MB, neuki se takoj začnejo pritoževati zakaj ne prenese dokument v 10 sekundah?! Če dobro pogledamo je podatek v MB kar pomeni 100 Mega Baytov, pomeni da je velikost v bitih praktično 800 Mb, tako če imamo prenos 10 Mb/s bomo datoteko prenašali 80 sekund in ne 10 sekund.

3.5.9 Kako merimo količino podatkov in kako hitrost prenosa podatkov? Kako predstavljajo kilobyte, megabyte, gigabyte, terabyte in petabyte?

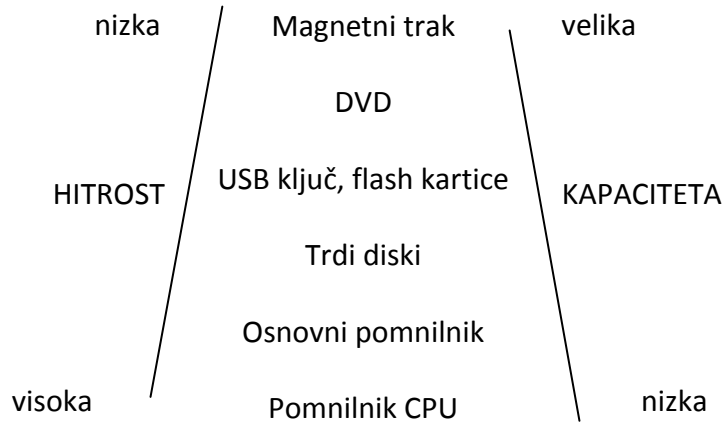
Količino podatkov merimo v bytih (B) ter hitrost prenosa v bitih na sekundo (b/s).

- kB – kilobyte – 2^{10} bytov
- MB – megabyte – 2^{20} bytov
- GB – gigabyte – 2^{30} bytov
- TB – terabyte – 2^{40} bytov
- PB – petabyte – 2^{50} bytov

3.5.10 Kakšne vrste pomnilnikov poznamo in kako jih razvrščamo?

Poznamo več vrst pomnilnikov:
- Notranji,
- zunanji,
    - način zapisovanja
    - zamenljivost pomnilnega medija
    - oblika dostopa

V glavnem jih pa delimo po hitrosti in kapaciteti:



Slika 40 Razvrstitev pomnilnikov (vir: G. Nikolić, 2012)

3.5.11 Kaj so razširitvene reže in vmesniške kartice?

Na matični plošči računalnika so navadni razširitvene reže(ang. expansion slots) kamor natikamo vmesniške kartice (adapter cards).
Nekaj razširitvenih rež in vmesniških kartic:
- Flash Memory Cards
    - Nudi možnost prenosa podatkov iz mobilnih ter drugih naprav,
- PCMCIA
    - Namenjena prenosnikom, dodaja spomin, zvok, fax/modem komunikacijo.
- Express Card modul
    - So novejše in zamenjujejo PCMCIA kartice. Zahtevajo manj energije ter so hitrejše.

3.5.12 Opišite in obrazložite vsaj pet vrat in konektorjev (ports and connectors)?

Poznamo več vrst vrat in konektorjev:

- USB (Universal Serial Bus), konektor in vrata preko katerih komunicira večina današnjih naprav. Poznamo še Mini USB in Mikro USB.
- LPT port, starejša vrata preko katerih so delovali večinoma tiskalniki, novejši računalniki več ne vsebujejo teh vrat.
- Audio Jack, priključek za zvok.
- Microphone Jack, priključek za mikrofon.
- RS232, priključek za serijska vrata, dosti dodatnih naprav se uspešno povezuje s tem priključkom saj ima tako imenovana priključka RX (Receive) in TX (Transmit) za pošiljanje in prejemanje podatkov.
- PS2 – miška, tipkovnica
- Firewire port, priključek za video od podjetja Apple.

3.5.13 Kaj so USB/Firewire/DVI/HDMI/S-video konektorji in kje jih uporabljamo?

USB priključek ali ang. Universal Serial Bus je priključek s katerim povezujemo večino današnjih periferij. Ima 4 priključke od tega dva napajalna +5 V ter dva za pošiljanje in prejemanje podatkov.
Firewire je priključek za prenos video podatkov podjetja Apple, zmore visoke hitrosti do 400 Mb/s.
DVI priključek ali ang. Digital Visual Inteface, digitalni priključek za zaslone.
HDMI priključek ali ang. High-Definition Multimedia Interface, vmesnik za visoko zmogljive video podatke oz. video podatke visoke kvalitete.
S-video priključek ali ang. Super Video, je priključek za prenos video signala na televizijo.

3.5.14 Na kaj vse moramo paziti, ko kupujemo računalnik?

Pri nakupu računalnika v večini primerov je kvaliteta in zmogljivost linearno odvisna od cene. Ob nakupi računalnika moramo najprej vedeti za kaj ga bomo uporabljali, ko smo odločeni za nakup moramo paziti, da kupimo ustreznega glede na uporabo. Na primer programer, ki bo pisal programe potrebuje zelo hiter računalnik (npr. za simulacijo programa in prevajanje) a ne potrebuje dobrega vizualnega vmesnika, za razliko od fotografa ali video obdelovalca, kateri pa potrebujejo visoke zmogljivosti grafike.

3.5.15 Opišite vrste računalnikov!

Poznamo:

- Mikroračunalniki
  - Majhni in namenjeni individualni uporabi PC (Personal Computer)
- Miniračunalniki
  - Zmogljivejši od PC, namenjeni zahtevnejšim nalogam.
- Veliki računalniki
  - Za vpisovanje, vnašanje in hranjenje velike količine podatkov.
- Super računalniki
  - So izredno hitri računalniki, za računanje zahtevnih računov, velikih matrik – napovedovanje vremena.

3.6 Vhodno/izhodne enote

3.6.1 Naštejte in natančneje opišite vsaj tri vhodne naprave!

Splošno so vhodne naprave, naprave, ki nam omogočajo pošiljanje podatkov na računalnik. Primer:

- Tipkovnica, s tipkanjem po tipkovnici pošiljamo niz števil na računalnik, kateri si tolmači določeno število kot simbol, ki se nam prikaže na zaslonu. Najbolj uporabljena tipkovnica je tipa QWERTY ali QWERTZ, razlika med tipoma je postavitev črke Y in Z.
- Biometrični bralnik prstnega odtisa. Lahko ga uporabimo kot ključ oz. geslo, vendar se je izkazalo, da niso najbolj zanesljivi in jih je mogoče hitro pretentati.
- Optični bralnik ali ang. scanner, je naprava katera prebere sliko ali besedilo oz. kar je na površini bralnika in pretvori v digitalni format na računalnik.
- Kamera, zaznava okolico in pošilja sliko na računalnik.
- Digitalni osciloskop, meritve, ki jih opravljamo z osciloskopom, lahko prenašamo na računalnik.

3.6.2 Naštejte in natančneje opišite vsaj tri izhodne naprave!

Izhodne naprave, so naprave na katere pošiljamo podatke iz računalnika. Primer:

- Tiskalnik, s pomočjo računalnika pošljemo tiskalniku sliko ali besedilo v obliki strojnega jezika kateri natisne želen dokument.
- Zaslon, monitor, prikazuje sliko dogajanja na računalniku, t.i. namizje računalnika.
- Zvočni kino oz. več-sistemski zvočni sistem, kateri nam omogoča, da iz računalnika pošiljamo nanj več vrst zvočnega signala s katerim tvorimo sistemski zvok.
- Plotter, XY risalnik, preko računalnika lahko izrisujemo razne grafe preko risalnika.

3.6.3 Katere biometrične vhodne naprave poznamo?

Vhodne biometrične naprave delimo na fiziološka in vedenjska in sicer:

Fiziološka:

- Obraz,
- prstni odtis,
- šarenica,
- roka,
- DNA.

Vedenjska:

- Način tipkanja,
- podpisovanje,
- glas.

3.6.4 Kateri so problemi pri uvajanju biometričnih naprav?

Največji problemi pri uvajanju biometričnih naprav so zanesljivost naprav in varnost. Prstni odtis se je izkazal za ne preveč zanesljivega, saj lahko odtis iz neke površine preslikamo in tako pretentamo bralnik. Nadzorno iskalne kamere, dobro zaznavajo obraze na kratkih razdaljah a osebe se lahko maskirajo.

3.6.5 Kaj so tipkovnice, kako jih delimo in kakšna je razlika med QWERTY in QWERTZ tipkovnicami? Katere druge tipe tipkovnic še poznamo?

Tipkovnica je skupek tipk, ki se uporabljajo skupno ali posamezno. Poznamo diskretne vhodne naprave kjer vstavimo eno ali dve diskretni poziciji (tipke na tipkovnici,…) ter zvezne vhodne naprave kjer vstavimo zvezno povezane podatke (pisala digitalne table, …).
Tipkovnice delimo na elektromehanske, to so vsakdanje tipkovnice katerih vrlina je dober odziv in občutek za vnos, ter membranske, katere so ravne, niso občutljive na prah, tekočine, ter primerne za velika farme in javne prostore.
Najbolj razširjene tipkovnice so tipa QWERTY in QWERTZ, če pogledate na tipkovnico je ime sestavljeni iz prvih črk tipkovnice vse do črke Y oz. Z, kar je ravno razlika teh dveh tipov tipkovnice – v postavitvi črke Y oz. Z.
Poznamo več vrst tipkovnic, kot so; virtualne tipkovnice, katere se projicirajo, akordne tipkovnice, t.i. natural elite tipkovnice, itn.

3.6.6 Kaj so lokatorji in posebej opišite vsaj tri lokatorje!

Lokatorji ali ang. pointing devices uporabljamo za določitev točke ali poti v eno ali dvo oz. tridimenzionalnem prostoru. Običajno so zvezne vhodne naprave z izjemo miške, ki združuje tako zvezne, kot diskretne pomike.
Zasloni občutljivi na dotik, lahko so folijski ali kapacitivni. Ob dotiku na površino, zazna dotik kjer se spremeni kapacitivnost oz. upornost in tako locira pozicijo, s tem premaknemo kazalec miške ali opravimo klik
Prostorska miška, z njo manipuliramo z objekti v tri dimenzionalnem (3D) prostoru.
Podatkovne rokavice ali ang. Dataglove katere pošiljajo v računalnik pozicijo roke in prstov.

3.6.7 Katere vrste zaslonov najpogosteje uporabljamo?

Najbolj pogosto uporabljeni zasloni so bili CRT (Chatode Ray Tube) katere so zamenjali t.i. LCD (Liquid Crystal Display). Na trgu so tudi že LED (Light Emitting Diode) zasloni.

3.6.8 Kakšna je razlika med prikazovalniki, kot so: (Opomba: podani bodo na izpitu)?

Poznamo alfanumerične, digitalne ter barvne prikazovalnike.
Alfanumerični prikazovalniki so pogosto pri opazovanju in vodenju sistemov, največkrat izvedeni mehansko z vrednostno skalo in kazalcem. Poznamo krožno, ločno, vertikalno, polkrožno in horizontalno skalo. Med te prikazovalnike štejemo tudi pisalnike s peresi, kot so ploterji, ipd.
Digitalni prikazovalniki so lažje berljivi z večje razdalje kot alfanumerični, vendar jih slabše zaznavamo, na primer merilec hitrosti v avtomobilu. Ob premikanju kazalca dobimo takoj občutek s kakšno hitrostjo vozimo za razliko ob skoku številk, kjer pa ta občutek izgubimo.

Barvni prikazovalniki so predvsem pomembni za opozorila in obvestila, imajo veliko vlogo pri zaznavanju nevarnih območij. Poznamo enobarvne, raznobarvne in barvna stikala.

3.6.9 Opišite 3D interaktivno animacija?

3D ali tridimenzionalna animacija nam daje občutek realnosti in prostorskost na ravni površini. Ker je na planetu le en izvor svetlobe (Sonce) se možgani ljudi orientirajo predvsem na senco, kateri zaupajo, tako lahko s senco manipuliramo z možgansko predstavitvijo okolice. 3D animacija se poslužuje ravno tega, da objekte senči in jih riše v tridimenzionalnem okolju, tako dobimo občutek prostora, spreminja prav tako kot opazovanja, s tem tudi senco, svetlobni izvor in perspektivne efekte.

3.6.10 Opišite zakaj nastopajo še vedno problemi pri zaznavi človeškega govora in kako se danes zajema govor v računalnik?

Zvok katerega narišemo v časovni diagram ima zvezno obliko, katero računalnik preko A/D pretvornika pretvori v digitalen signal. Bolj dober sample rate ima A/D pretvornik, bolje je signal pretvorjen in se bolj ujema z realnim, toda še vedno prihaja do odstopanj, kar povzroča težave pri zaznavi zvoka.

3.6.11 Kaj je skener in zakaj ga lahko uporabljamo?

Skener je optični bralnik. Uporabljamo ga za optično branje dokumentov. Navadno sestoji iz ohišja s stekleno stranico kamor položimo dokument katerega želimo prebrati ter pokrova z belim ozadjem.

3.6.12 Kaj je MIDI zvočni sintetizator ter kako in kje ga uporabljamo?

MIDI (Musical Instrument Digital Interface) je elektronski standardni protokol, s katerim lahko komunicirajo različne elektronske glasbene naprave. Uporabljamo ga za prenos signala iz elektronskega glasbila na računalnik ter obratno. MIDI kot že povedano, je protokol in ne vsebuje zvočnih zapisov, temveč sprejema podatke in jih pošilja naprej v točno določene tone. Standardni MIDI ima konektor DIN5 kateri ima 3 priključke za pošiljanje informacij ter 2 za razširitev.

3.6.13 Kaj so senzorji in katere senzorje uporabljamo v elektrotehniki?

Senzorji so naprave, katere pretvarjajo fizikalno veličino v električno. Primer senzorjev, ki jih uporabljamo so; senzor pritiska, vlage, osvetljenosti, toplote, obratov, senzor natezne sile, senzor zvoka – mikrofon, gibanja, itd.

3.6.14 Kaj so force feedback naprave? Navedite primer njihove uporabe?

Force feedback naprave so naprave katere tudi vračajo podatke. Primer take naprave je joystick, ko igramo neko igro ter se zgodi nek dogodek na katerega nas želi joystick opozoriti, navadno to stori z vgrajenim vibratorjem kateri zavibrira joystick in nam na ta način da vedeti.

3.6.15 Kakšna je razlika med analognimi in digitalnimi signali?

Analogni signali so za razliko od digitalnih signalov največkrat zvezni in neprekinjeni, pri digitalnih vrednostih pa so funkcije največkrat stopničaste, jih zapišemo s številkami. Tak signal tipamo.

3.6.16 Opišite vsaj tri prikazovalnike za avtomatizacijo procesov!

Frekvenčni pretvornik za regulacijo obratov motorja, kateri ima prikazovalnik frekvence in števila obratov motorja.
Avtomatske kontrolne tehtnice za tehtanje napolnjenosti vreč s prikazovalnikom teže.
Merjenje porabe toplotnega ogrevanja z brezžičnim prenosom podatkov in prikazom na prenosni napravi.
Polnjenje in praznjenje cisterne z gorivom ter prikaz nivoja goriva v cisterni.
Tekoči trak za razvrščanje in izmet slabih izdelkov s prikazom razvrščenih izdelkov ter števila izmetov.

3.6.17 Kako zajemamo zvok/video?

Zvok zajemamo z različnimi vrstami mikrofonov, kateri so povezani na računalnik ali pred tem še na predojačevalnik. Ko govorimo v zvočnik se tresljaji pretvarjajo v električne signale, katere zajema računalnik.
Video lahko zajemamo z najrazličnejšimi kamerami, lahko so samostojne kamere z lastnim pomnilnikom ter nato prenesemo film oz. video na računalnik, lahko pa je kamera povezana z računalnikom, kamor se direktno prenaša posnetek.

3.6.18 Katere standarde poznamo za video obdelavo in zakaj potrebujemo kompresirani video?

Poznamo PAL (Phase Alternating Line) standard s prenosom 25 sličic na sekundo. Primer: slika 640 x 480, 25 slik/sek, 8 bitov na točko je hitrost prenosa podatkov;

$$640 \cdot 480 = 307200 \text{ pikslov}$$
$$307200 \cdot 8 = 2457600 \text{ bitov}$$
$$2457600 \cdot 25 = 61.440.000 \text{ bitov/sekundo}$$

Hitrost prenosa je ~60 Mb/s.

Kompresiranje videa potrebujemo za zmanjšanje datotek za prenos, s tem pridobimo na prenosu in prostoru, ki ga te datoteke zavzemajo.

3.6.19 Kakšna je razlika med DLP/CRT/OLED/SED/LCD/plasma zasloni?

- DLP – Digital Light Processing, s svetlobo preko barvnega filtra ter nekaj optike, osvetljuje DMD (Digital Micro-Mirror Device) napravo katera preko dodatne optike projicira sliko.
- CRT – Chatore Ray Tube, ekran s pomočjo izvora snopa elektronov tvori na površini ekrana, kateri je steklen, sliko. Značilnost teh zaslonov je, da so bili veliki in težki.
- OLED – Organic Light Emitting Diodes, organski elementi reagirajo na električni tok tako, da pričnejo emitirati neko svetlobo, katero mi izkoriščamo. Dobra lstnost teh je, da so fleksibilni, jih lahko ukrivimo.
- SED – Surface-conduction Electron-emitter Display, deluje na enakem principu kot CRT, torej ima izvor elektronov kateri tvorijo sliko na zaslonu. Razlika me SED in CRT je, da ima SED več matrik izvora elektronov, kar omogoča, da je zaslon bistveno tanjši.
- LCD – Liquid Crystal Display, danes najbolj razširjen prikazovalnik, saj je že skoraj v vsaki napravi. Za razliko od predhodnih omenjenih ima manjšo porabo, je tanjši in se bistveno manj segreva. Edina težava LCD prikazovalnikov je vidni kot, česar pri CRT ni, vendar se je tudi ta kot že bistveno bolj izboljšal.

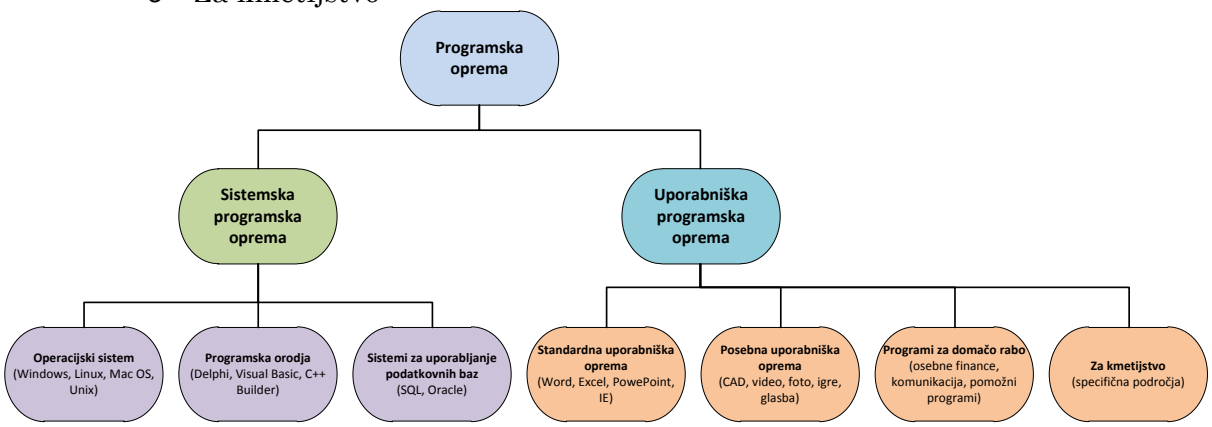
Vsi našteti prikazovalniki imajo eno skupno lastnost – da prikazujejo sliko na površino. Razlik med njimi pa je nešteto, od porabe, do velikosti, teže, načina delovanja in uporabnosti.

3.7 Programska oprema

3.7.1 Narišite in opišite skico razdelitve programske opreme!

Programsko opremo delimo v hierarhično zgradbo:
Programska oprema

- **Sistemska programska oprema**
  - Operacijski sistemi
  - Programska orodja
  - Sistemi za upravljanje podatkovnih baz
- **Uporabniška programska oprema**
  - Standardna uporabniška oprema
  - Posebna uporabniška oprema
  - Programi za domačo rabo
  - Za kmetijstvo



Slika 41 Razdelitev programske opreme (vir: G. Nikolić, 2012)

3.7.2 Naštejte in opišite vsaj štiri primere programov za: poslovne II grafične in multimedijske II izobraževalne II komunikacijske II pomožne programe!

Poslovni programi:
- Za obdelovanje besedil (Microsoft Office Word)
- Za obdelovanje tabel (Microsoft Office Excel)
- Za podatkovne baze (SQL)
- Za arhiviranje dokumentov (ZIP, WinRar)

Grafični in multimedijski programi:
- Računsko vodeno načrtovanje (CAD)
- Obdelava slik (Adobe Photoshop)
- Video in zvočna obdelava (Adobe Premiere)
- Orodja za izdelavo spletnih strani (Web Expression)

Domači, izobraževalni programi:
- Načrtovanje doma, vrta (Google Sketch Up)
- Dohodnina (program za izračun dohodnine)
- Za obdelavo prezentacij (Microsoft Office Power Point)
- Programi za obdelavo zvoka in videa (Windows Movie Maker)

Komunikacijski programi:
- E-pošta (Gmail, Outlook)
- Klepetalnice (Skype, MSN Messenger)
- Videokonference (Skype)
- Programi za prenos podatkov (FTP – File Transfer Protocol)

Pomožni programi:
- Antivirusni programi (Microsoft Security Essentials)
- Programi za kompresijo (ZIP, RAR)
- Požarni zidovi (Windows Firewall)
- Predvajalniki (WMP)

3.7.3 Kakšne so razlike med licenčnimi programi, odprto kodnimi programi, shareware, freeware in public domain programi?

Za licenčne programe potrebujemo licenco za delovanje, katera pa je plačljiva. Odprtokodni so za razliko od licenčnih brezplačni in jih lahko urejamo, popravljamo in prosto dajemo naprej. Shareware so programi, ki so avtorsko zaščiteni in se pogosto prosto uporabljajo le za določen čas. Freeware so prav tako kot shareware avtorsko zaščiteni, toda ti se brezplačno uporabljajo. Public-domain software, kot že ime pove so programi, kateri niso avtorsko zaščiteni ter so javno dostopni.

3.7.4 Naštejte in opišite vsaj štiri programe za vzdrževanje računalnika in integrirana okolja!

Programi za vzdrževanje računalnika in integrirana okolja:
- CCleaner – program za pregled in čiščenje starih začasnih datotek ter obnovo registrov
- Revo Uninstaller – program za brisanje programov
- IObit Uninstaller – omogoča brisanje programov, ki se jih ne da z že vgrajenim orodjem v okolje Windows.
- AppSnap, QwinApt, Techtracker – programi za avtomatske nadgradnje programov.
- Explorer – za dosego datotek oz. lokacij na računalniku

3.7.5 Kakšna je vloga programov za urejanje dokumentov in kaj nudijo?

Vloga programov za urejanje dokumentov so v večini primerov pisni dokumenti v obliki besedila, slike, računi, načrti, ipd. Nudijo nam; pisanje, avtomatsko popravljanje besed, črkovanje, vključevanje slik, dodajanje opomb, vključevanje grafov, tabel, enačb, vključevanje makrojev, avtomatsko generiranje seznamov in kazal, ipd.

3.7.6 Naštejte in opišite vrste programov za urejanje dokumentov!

Poznamo editorje za vnos besedil, urejevalnike besedil, ki prav tako omogočajo vnos besedila ter tudi nekaj oblikovanja, programe za namizno založništvo, ki imajo šibkejše sposobnosti vnašanja besedil, so pa močnejši na področju oblikovanja, ter programi za oblikovanje besedil, so pa programi, ki oblikujejo besedilo, ki je napisano z besedilom in posebnimi ukazi za oblikovanje.

3.7.7 Kakšna je razlika med editorji II urejevalniki besedil II programi za namizno založništvo II oblikovalci besedil II orodji za tvorbo spletnih strani ?

Editorji:
- So programi za vnos preprostih besedil. Besedilo je sestavljeno samo iz znakov in ne vsebuje nobenega oblikovanja.
    - Eden najbolj uporabljenih je zagotovo beležnica ali ang. Notepad.

Urejevalniki besedil:
- So prav tako programi za vnos besedil kot editorji, le da tukaj besedilo ni sestavljeno zgolj iz znakov, temveč vsebuje tudi oblikovanje besedila, številčenje, vgradnja lik ipd.
    - Zelo pogosto uporabljen program je MS Office Word.

Programi za namizno založništvo:
- Ti programi imajo šibkejše funkcije za vnašanje in urejanje besedil a so zelo močni na področju za oblikovanje besedila. Primerni so za oblikovanje pred tiskanjem.
    - Primer programa bi bila Corel Ventura in Adobe Pagemaker.

Oblikovalci besedil:
- So programi, ki oblikujejo besedilo, ki je napisano z besedilom in posebnimi ukazi za oblikovanje.
    - Primer programa bi bil TeX, Runoff in FrontPage (za HTML).

3.7.8 Kakšna je vloga in funkcija programov za delo s tabelami?

Programi za delo s tabelami rešujejo marsikatero težavo inženirskih in finančno knjigovodskih problemov. Numerične vrednosti organizirajo v stolpce in vrstice, definiramo lahko raznorazne formule za izračun, poleg izračunov, pa imajo možnost še grafične predstavitve rezultatov.
Eden mnogih programov je precej močno orodje in sicer Microsoft Office Excel, v katerem lahko definiramo kopico različnih formul in predstavitev. Primer funkcije katera bo preštela število celic pod pogojem, da je število v celici večje od 25:

$$= countif(B1 : B10;" > 25")$$

3.7.9 Kakšna je vloga in funkcija programov za delo s podatkovnimi bazami?

Vloga programov za delo s podatkovnimi bazami je urejanje velike količine podatkov, kateri se velikokrat vnašajo avtomatsko in ne le ročno. Imajo funkcijo povezovanja podatkov med različnimi datotekami, vzpostavljajo relacije. Primer takšnih programov so; MS Access, Dbase, paradox, …

3.7.10 Kaj pomeni izraz "relacija" na področju podatkovnih baz in kako jo uporabljamo?

Relacija je nek odnos, zveza, torej ima pri podatkih enak pomen. Vzpostavljanje relacij med datotekami pomeni, da se v dveh različnih datotekah na različnih krajih nahaja kratica oz. neka označba (npr. zaporedna št. banke), nahaja se v obeh datotekah in se na ta način vzpostavi relacija.

3.7.11 Katere jezike in programe uporabljamo za podatkovne baze? Kaj je njihova značilnost?

Programi za urejanje zbirke, baze podatkov imajo svoj programski jezik, primeren za delo z datotekami. Poznamo SQL – Structured Query Language programski jezik, dBASE ipd. Značilnost programov za podatkovne baze je, da organizirajo podatke v tabele, lahko je vse v eni tabeli vendar zaradi podvajanja podatkov dodajamo ID naslove in ustvarjamo več tabel katere medsebojno relacijsko povezujemo.

3.7.12 Kako se delijo enote v podatkovnih bazah? Opišite te enote in navedite primer za vsako enoto!

Enote delimo na:

- Bazo podatkov (database);
  - Je skupina datotek, ki skupaj nekaj opisujejo, npr.: evidenca študentov.
- Datoteke (data file);
  - So lahko računske simulacije, kartoteke, sestavljene iz formularjev. Vsebujejo podatke, npr.: seznam študentov prijavljenih na dodatne vaje.
- Zapis (record);
  - Kartica v kartoteki, katera vsebuje okenca in polja za zapis podatkov, npr.: podatki o študentu.

- Polje (field);
    - Okence na kartici, v katero vpišemo zgolj eno vrednost, lahko je numerična, znakovna ipd. npr.: ime študenta, priimek, ipd.

3.7.13 Kakšna je vloga in funkcija programov za vstavljanje opomb! Navedite primere!

Programe za vstavljanje opomb uporabljamo, da že obstoječim dokumentom pripišemo ali pripnemo slikovno ali tekstovno opombo. Na primer Microsoft Office Word in možnost dodajanja sprotnih opomb k besedilu.

3.7.14 Kakšna je vloga in funkcija programov za prezentacije! Navedite primere!

Vsaka boljša predstavitev naj si bo to izdelka, projekta ali česarkoli drugega vsebuje vizualno predstavitev informacij o slednjem. V ta namen uporabljamo programe za pripravo prezentacij. Eden zelo razširjenih programov je zagotovo Microsoft Office Power Point.

3.7.15 Kakšna je vloga in funkcija programov za matematiko! Navedite primere!

Vsak študent se je srečal že z več računskimi problemi, na primer matrika. Računanje malih matrik ni problem, malce večje pa nam že predstavljajo težavo za računanje, da ne omenjamo velikih matrik. Tudi funkcije si nekako težko predstavljamo, ko so zapisane z enačbo, v ta namen uporabljamo matematične programe, kateri nam rešujejo težje oz. zapletene račune ter nam grafično prikazujejo obliko funkcij. Primer takšnih programov je zagotovo Matlab, Graph, Scilab, Mathematica, ipd.

3.7.16 Kakšna je vloga in funkcija programov za risanje! Navedite primere!

Programi za risanje, kot že samo ime pove so za izdelavo skic, risb, načrtov. Poznamo več vrst programov za risanje. Od manj zahtevnih, kjer rišemo ploskovno torej dvodimenzionalno (2D) do malce težjih, kjer lahko rišemo v treh dimenzijah (3D).
Primeri programov bi bili: paint programi, MS PaintBrush, Photodhop, draw programi, CorelDraw, programi za poslovno grafiko, Excel, Charisma ter CAD programi.

3.7.17 Kakšna je razlika med bitno in vektorsko sliko?

Vektorska slika kot že ime pove je sestavljena iz vektorjev (črt in krivulj). Vsaka črta ali krivulja je lahko sestavljena iz več vozlišč z ravnimi odseki, predmete pa lahko zapolnimo z barvo ali vzorcem. Dobra lastnost vektorskih slik je, da se ob povečavi kvaliteta ne uniči.
Bitna slika kot ime pove je sestavljena z biti-pikami. Vsaki piki je dodeljena lokacija in barvna vrednost. Slabost bitnih slik je, da se ob povečavi kvaliteta kvari, saj se povečujejo pike, ki tvorijo sliko.

3.7.18 Kakšna je vloga in funkcija programov za zvočno zajemanje! Navedite primere!

Poznamo več vrst programov za zvočno zajemanje glede na njihovo funkcionalnost. Programi za zajemanje zvoka kot glasbe, se uporabljajo za tvorjenje glasbenih datotek, ipd. Poznamo pa programe za razumevanje izgovorjenih besed ang. voice recognition.

Takšni programi se že pojavljajo v Windows okolju, njegovo ime je Speech Recognition in sicer je za krmiljenje na glasovni ukaz. Z glasom usmerjamo računalnik, kateri se tudi lahko sproti uči besed, ima pa tudi že nek majhen nabor izgovorjenih besed.
These sentence was written by a speech recognition.

3.7.19 Kje so problemi pri zvočnem zajemanju govora? Katera dva načina za zaznavanje sposobnosti izgovorjene besede poznamo?

Pri zvočnem zajemanju govora so predvsem pri razločevanju izgovorjenih besed. Da nas program razume dobro moramo dobro poznati izgovarjavo besed, biti v mirnem okolju, torej ne sme biti okrog nas kakršnihkoli drugih zvokov, ter dobro moramo govoriti jezik, katerega razume program. Poznamo dva načina zajemanja izgovorjenih besed; da program naučimo zaznavati lastne izgovorjene besede ter, da ima program že vključen nabor izgovorjenih besed.

3.7.20 Kakšna je vloga in funkcija programov za video zajemanje! Navedite primere!

Programi za zajemanja in obdelavo nudijo zajemanje in obdelavo video segmentov. Poznamo več programov, kot so: Adobe Premiere, Camtasia Studio, Windows Movie Maker.

3.7.21 Kakšna je vloga in funkcija programov za projektno vodenje! Navedite primere!

Programi za projektno vodenje nudijo načrtovanje, planiranje, analizo, spremljanje virov in stroškov projekta. Primer programa je MS Project.

3.7.22 Kakšna je vloga in funkcija programov za osebno informiranje! Navedite primere!

Programi za osebno informiranje ali osebni informatorji (Personal Information MAnager) so programi za delo z dlančniki ali pametnimi telefoni. Primeri takšnih aplikacij so raznorazni koledarji, beležnice, programi za sinhronizacijo, ipd.

3.7.23 Navedite in opišite vsaj štiri internetna orodja!

Raznorazni brskalniki; Mozilla FireFox, Google Chrome, Internet Explorer. Omogočajo nam dostopati do raznoraznih spletnih dokumentov, strani.
Komunikacijska orodja; MSN Messenger, Skype, nam omogočajo neposredni pogovor, video ali tekstovno.
Omrežna orodja; FTP in omrežni analizatorji, omogočajo nam prenos datotek na splet.
Kompresijska orodja: WinZip, s pomočjo teh orodij kompresiramo datoteke, da je nijihova končna velikost manjša.

3.7.24 Kako naredimo 3D fotografske slike?

3D fotografske slike lahko naredimo na več načinov. Eden izmed načinov je, da objekt fotografiramo, nato pa se zamaknemo le za malenkost (fotografi pravijo, da je dovolj velik zamik že, če težo prenesemo na drugo nogo) in naredimo še eno sliko. Taki dve fotografiji

nato postavimo eno ob drugo ter se poizkusimo zazreti skoznjo (enako kot pri stereogramih), z malo truda nam uspe in sliki se nam združita v 3D podobo. Podobno lahko naredimo z obdelavo slike ter nato uporabimo očala s katerimi vidimo 3D podobo, ki pa je bistveno manj očitna kot pri stereogramih.

3.7.25 Kako naredimo panoramske fotografske slike?

Za panoramsko sliko moramo zajeti večjo število fotografij v okolici. Najbolje je to početi s stativom, kateri je nepremičen, nato pa zajemamo slike v krogu. Nato s programom za panoramske slike (po navadi dobimo tak program že ob nakupu fotoaparata) združimo fotografije. Primer panoramske slike, katere avtor sem sam:

Slika 42 Panoramska slika, Pohorje, Razgledni stolp (vir: G. Nikolić, 2010)

3.7.26 Na kaj vse moramo paziti pri nakupu programske opreme?

Ob nakupi programske opreme moramo biti pozorni na to ali se sklada s specifikacijami našega računalnika. V mislih imamo predvsem 32 in 64 bitni sistem. Ko kupujemo programsko opremo pazimo, da so vključene ob nakupu tudi posodobitve programske opreme ali ang. updates, v nasprotnem primeru bo programska oprema prej kot slej zastarela.

3.7.27 Kaj pomenijo izrazi: PIM / CAD / SQL / Open Source / HTML / XML in kje ter v kakšne namene jih uporabljamo?

PIM – Personal Information Manager, osebni informator, skrbi za lstno informiranje, to so programi oz. dodatki za računalnik ali pametni telefon; koledar, beležnica, …
CAD – Computer-aided design, računalniško podprto risanje tehnične dokumentacije.
SQL – Structed Query Language, programski jezik za urejanje zbirke oz. baze podatkov.
Open Source – so odprtokodni programi, katere lahko prosto uporabljamo, popravljamo in posredujemo naprej.
HTML – HyperText Markup Language – programski jezik za programiranje spletnih strani.
XML – Extensible Markup Language, enako kot HTML, je programski jezik za programiranje spletnih strani. Razlika je v tem, da HTML podatke oblikuje in prikazuje XML pa definira kjer podatki so in jih opisuje.

3.8 Vprašanja in odgovori iz kvizov za samostojno preverjanje znanja

3.8.1 Programiranje v programskem jeziku C

3.8.1.1 Kateri od predznakov se ne uporablja za kazalce?

a. Znak &
b. Znak #
c. Znak *

3.8.1.2 Funkcijo določimo na začetku na naslednji način: function (a,b); .

Odgovor: Drži **Ne drži**

3.8.1.3 (vstavi manjkajočo besedo) Programski modul v jeziku C se imenuje (v originalu) _____ .

Odgovor: **function**

3.8.1.4 Strukturo definiramo s stavkom "structi Osebni_podatki oseba".

Odgovor: Drži **Ne drži**

3.8.1.5 Z "enum" ukazom izdelamo oštevičeni tip spremenljivk.

Odgovor: **Drži** Ne drži

3.8.1.6 Kateri izmed izrazov ni polje ali matrika?

a. p [2][5]
b. p [2]
c. p [2],[5]

3.8.1.7 S kazalci ne moremo kazati na funkcije.

Odgovor: Drži **Ne drži**

3.8.1.8 Polje znakov na koncu besede vsebuje znak "\0" .

Odgovor: **Drži** Ne drži

3.8.1.9 (vstavi manjkajočo besedo) Izraz _____ se uporablja zato, da povemo, da funkcija ne vrne nobene vrednosti.

Odgovor: **void**

3.8.2 Osebni računalnik (strojna oprema)

3.8.2.1 Katera naprava ne spada med vhodne naprave?

a. SED zaslon
b. Podatkovna roka
c. Tipkovnica
d. Zaslon, občutljiv na dotik
e. Miselni sledilec

3.8.2.2 Koliko je 1 Kbyte?

a. 1.024 bytov
b. 1000 bytov
c. 1.024 bitov

3.8.2.3 Super računalniki so sposobni dosegati hitrosti večje od?

a. PFLOPS
b. PHz
c. TFLOPS

3.8.2.4 Kateri od navedenih ne spada med vrste notranjih pomnilnikov?

a. RAM
b. EEPROM
c. PIROM
d. ROM

3.8.2.5 Kaj dela aritmetično logična enota?

a. V njej se izvajajo logične in aritmetične operacije
b. V njej se izvajajo logistične in matematične operacije
c. V njej se izvajajo seštevalne operacije

3.8.2.6 Katera naprava ne spada med izhodne naprave?

a. Tiskalnik
b. Panel
c. Skener
d. DLP zaslon

3.8.2.7 Kateri stavek kaže na pravilni vrstni red glede hitrosti pomnilnikov?

a. trdi disk, CPU pomnilnik, USB ključ in magnetni trak
b. CPU pomnilnik, trdi disk, USB ključ in magnetni trak
c. CPU pomnilnik, USB ključ, trdi disk in magnetni trak

3.8.2.8 Kateri priključek ne spada med priključke za zaslon?

a. Ethernet
b. VGA
c. HDMI
d. DVI

3.8.2.9 Kaj so in kako delujejo "dual-core" procesorji?

a. So čipi, ki imajo dva procesorja in delujejo hkrati
b. So čipi, ki vsebujeta dva procesorja in delujejo po principu prepletanja
c. So čipi, ki vsebujejo dva različna jedra procesorja in delujejo po principu prepletanja

3.8.2.10 Kaj ni del centralno procesne enote?

a. Krmilna enota
b. Zunanji pomnilnik
c. Aritmetično logična enota
d. Notranji pomnilnik

3.8.3 Osebni računalnik (programska oprema)

3.8.3.1 Licenčne programe lahko uporabljamo brez plačila avtorju izdelka.

Odgovor: Drži **Ne drži**

3.8.3.2 Kateri program ne spada med internetna orodja?

a. HTML orodja
b. Orodja za elektronsko pošto
c. FTP orodja
d. CAD orodja

3.8.3.3 Program za tehnično risanje (CAD) omogoča risanje slik v vektorski obliki.

Odgovor: **Drži** Ne drži

3.8.3.4 Programi za projektno vodenje so namenjeni planiranju in spremljanju aktivnosti.

Odgovor: **Drži** Ne drži

3.8.3.5 Katera funkcija ne spada v programe za urejanje dokumentov?

a. Vključevanje tabele
b. Vektorsko risanje
c. Vključevanje slik
d. Pisanje

3.8.3.6 Kateri program uporablja bitno sliko?

a. CorelDraw
b. Adobe Photoshop
c. Excel

3.8.3.7 Kateri način ni pravilen za zvočno zajemanje in razumevanje izgovorjene besede?

a. Program ima že vnešen nabor izgovorjenih besed
b. Program naučimo zaznavati lastne izgovorjene besede
c. Računalnik ima odličen mikrofon in program za kvalitetno zajemanje zvoka.

3.8.3.8 Kaj ne spada v program za osebno informiranje?

a. Beležnica
b. Risar
c. Naslovi
d. Koledar

3.8.3.9 Kaj so programi za vzdrževanje računalnika in integrirana okolja?

a. So programi za organizacijo računalnika.
b. So programi za delo z računalnikom.
c. So programi za organizacijo datotek in podatkov, delovanje računalnika in za zaganjanje drugih programov.
d. So programi za delo z datotekami in podatki ter za zaganjanje operacijskega sistema.

3.8.3.10 Kaj so baze podatkov?

a. Je zbirka informacij skupaj s sistemom za razvrščanje, iskanje in urejanje teh informacij.
b. Je zbirka datotek, zapisov in polj.
c. Je zbirka dokumentov skupaj s sistemom za razvrščanje, iskanje in urejanje podatkov v bazi.

4. VIRI IN LITERATURA

Slika 36: Spletni vir, dostopno na URL:
http://en.wikipedia.org/wiki/International_Obfuscated_C_Code_Contest (April, 2012)

Prosojnice predavanj izr. prof. dr. DEBEVC MATJAŽ, univ.dipl.inž. el. ter kvizi za samostojno preverjanje znanja s strani https://estudij.uni-mb.si/