**HEAD PROGRAMMEERIMISTAVAD**

**Programmi kirjutame me üldjuhul üks kord, samas, juba kirjutatud koodi loeme korduvalt:**

* programmi silumisel,*

* programmikoodis vigade parandamisel,*

* muutuste tegemisel programmis,*

* programmikoodi uuendamisel,*

* programmikoodi uurimisel ja selle mõistmisel.*

**Kõik, mis me võtame ette, et muuta koodi loetavamaks, säästab hiljem hulga**

**väärtuslikku aega.** Võib juhtuda, et koodi peab hiljem muutma keegi, kes pole

selle autor ega sellega varem kokku puutunud. Kasutades ühtset lähenemisviisi

(standardit) programmikoodi loomisel, on kõigil lihtsam koodist aru saada.

Käesolev dokument käsitleb programmeerimisviise lähtudes keelest C/C++ ning

on orienteeritud ainetele IAG0581 Programmeerimine I ning IAG0582 Programmeerimine II.

Näidetes kasutatud kood põhineb samuti C/C++’l.

Edukat programmeerimist!

 **1 Koodi dokumenteerimine. Kommentaarid**

Iga programmikoodi sisaldav fail, mille me kirjutame, peaks algama

kommentaaridega, milles sisaldub selle faili enda nimi, faili autori nimi, faili

loomise kuupäev, viimase modifitseerimise kuupäev (ja kellaaeg) ning

lühikirjeldus all oleva programmikoodi eesmärkidest. Täiendavalt esitatakse

vajadusel kommentaarides ka eel- ja järel tingimused, tööks vajalikud

eeldused, mis võivad olla seotud antud programmifaili tööga.

Näiteks:

/\*

 FAILINIMI: main.cpp

 AUTOR: Anna Minna

 LOODUD: 20.09.2023

 MUUDETUD: 25.09.2023

 KIRJELDUS: See programm demonstreerib, kuidas ...

 Eeltingimused: ...

 Järel tingimused: ...

\*/

Lisaks programmifaili enda kommentaaridega varustamisele tuleks

kommentaaridega varustada ka kõik temas sisalduvad funktsioonid

(meetodid). Kommentaarid peaksid andma ülevaate sellest, mida antud

meetod teeb, millised on tema tööks vajalikud eeltingimused ja tööga

kaasnevad järel tingimused (tulemused). Samuti tuleks sisendparameetri(te) ja

väljundi olemasolul ka neid kirjeldada.

Näiteks:

/\*

 Arvutab vastavalt etteantud muutuja summa väärtusele

 konstandiga määratud kursi alusel väärtuse eurodes.

 Sisendparameetrid: summa: on summa kroonides

 Väljund: tagastab summa eurodes

\*/

double convertEekToEur( double summa )

{

 // Teisenda EEK’id EUR’ideks

 return convertEekToEur = summa / EUR\_RATE;

}

Kõiksugu täiendavate kommentaaride, märkuste tegemine programmikoodi,

muudab sellest arusaamise paremaks. Kommentaarid võivad eelneda

lauseteplokile või olla kirjutatud ploki/lause järele.

double keskHinne = 0.0F; // tudengi keskhinne õppeastmel X

int sooritusteArv = 10; // tudengi soorit. ainete koguarv

// väljasta kõik tudengi sooritatud õppeained

 while( i < sooritusteArv ) { ... i++; }

 **1.1 Treppimine**

Treppimine on programmi plokkidesse kirjutamise viis selle loetavuse

parandamiseks.

Soovitusi:

* loogiliselt kokkukuuluvate koodiridade eristamiseks kasuta nende vahel*

*ühte või kahte tühja rida*

* lisa alati tühi rida enne deklaratsioone*

* päisesse vormistatud deklaratsioonide puhul jäta ka peale*

*deklaratsioone tühi rida*

* enne ja pärast tingimus-ja tsüklilauset jäta 1 tühi rida*

* peale koma (,) lisa alati tühik*

* kõikide binaaroperaatorite (tehted) ette ja taha jäta 1 tühik*

* peale „avavat sulgu” ehk „{” alusta igat järgnevat programmirida nihkes*

*paremale (soovitavalt 3-4 tühikut) kuni „sulgeva suluni” ehk „}”*

* võimalusel väldi „tab” sümbolit treppimisel, kuna erinevad*

*tekstiredaktorid võivad seda kuvade erineva pikkusena.*

**2 Muutujate nimede vormistamine**

Üldiselt kehtib igas programmeerimiskeeles reegel, et muutujate nimed

algavad tähega, ning ei sisalda tühikut ega järgmisi sümboleid: \*, ^, #, +,

kusjuures, täiendavalt tuleks arvestada, et eesti keeles olemasolevate

täpitähtede kasutamine ei ole otstarbekas, kui mitte võimatu (vastuolus keele

reeglitega ning kompileerimisel antakse veateade). Mõningates keeltes on

lubatud alustada muutuja nime ka allkriipsuga.

**2.1 Suur- ja väiketähtede kasutamine**

Tähelepanu tasub pöörata ka suur- ja väiketähtede kasutamisele, kuna paljud

keeled on tõstutundlikud (case sensitive) sealhulgas ka C/C++, mis tähendab,

et suurtähe asendamine väiketähega või vastupidi, ei anna samaväärset

tulemust (tegu on siis kahe erineva muutujaga).

**2.2 Muutujate nimed**

Muutujatel peaksid olema nimed, mis väljendavad nende sisu. Näiteks, kui

soovime hoida muutujas kellegi eesnime, siis tasukski panna muutuja nimeks

eesnimi, mitte aga a, e1, vms., kuna viimased ei väljenda mitte mingil viisil

antud muutuja olemust. Muutujate nimedes kasutatakse ainult inglise

tähestiku suur ja väike tähti ning numbreid (a-z, A-Z, 0-9).

Erandiks siinkohal on tsüklimuutujad. Levinud praktika kohaselt märgistatakse

esimese tsükli muutujat tähega i, temas sisalduva teisese tsükli muutujat

tähega j, kolmandat k jne.

Näiteks:

int maatriks[10][20];

for( int i = 0; i < 10; i++ )

{

 for( int j = 0; j < 10; j++ )

 {

 // algväärtusta maatriks

 maatriks[i][j] = 0;

 }

}

Ühetähelisi lühendeid näiteks: x ja y on otstarbekas kasutada vaid siis, kui

nende nimi kannab üldtuntud teadmust, näiteks kui on tegemist koordinaatide

esitamisega.

Muutujatele nimede andmisel tuleks silmas pidada ka nende tüüpi, mis võiks

kajastuda ka nimes. See muudab koodi tunduvalt loetavamaks, kuna koheselt

on teada ka mis tüüpi muutujaga tegu on (vastasel juhul tuleks hakata kusagilt

otsima muutuja deklaratsiooni; või nuputama veidi, näiteks PHP puhul ei ole

muutujate deklareerimine vajalik). Juhul, kui muutujal tõesti ei ole kandvat

sisu, siis võib nimena kasutada ka tema tüüpi ning muutujad numereerida,

näiteks string1, string2, ... või kasutada lühikesi nimesid:

* s, t, u, or s1, s2 - kasutatakse tavaliselt stringide korral;*

* p, q, r, s - kasutatakse tavaliselt tõeväärsustüüpi muutujate*

*esitamiseks;*

* w, x, y, z – kasutatakse tavaliselt arvtüüpi muutujate jaoks;*

Mõningates vanemates keeltes on esitatud piirang, et muutuja nimi ei tohi

ületada 8 sümbolit. Sellisel juhul järgitakse reeglit, kus jäetakse ära vokaale,

alustades paremalt. Näiteks muutuja maxVolume kannaks sellise juhul nime

maxVolum. Paraku on sellisel kujul esitatud muutujate nimesid raske meelde

jätta ning neist ka raskem aru saada.

Parima nime valimine võib võtta veidi aega, kuid see tasub vaeva.

Programmeerimispraktikas on pikad muutujate nimed tavalised.

Parima nime valimine võib võtta veidi aega, kuid see tasub vaeva.

Programmeerimispraktikas on pikad muutujate nimed tavalised.

Mis aga juhtub siis, kui muutujale on antud tema sisu väljendav nimi, kuid ühel

hetkel peaks see muutuja hakkama sisaldama hoopis teistsugust

samatüübilist infot. Näiteks, eesnime asemel otsustame, et nüüdsest peaks

süsteemis kasutatama perenime või ees- ja perenimest koostatud

stringavaldist. Vastus oleks, et sellisel juhul peaks muutuma ka muutuja nimi.

Muutujate deklareerimisel peetakse aga heaks stiiliks kirjeldada iga muutuja

eraldi real, nagu tehtud alljärgnevs näites:

char\* username = „mina”;

char\* password = ”parool123”;

Selline ülestäheldusviis tagab parema koodi loetavuse.

**Muutujate nimede kirjapanekul tuleks järgida järgmisi reegleid:**

 muutuja nimi peaks olema nimisõna;

 pikkade muutujanimede (mitmest sõnast koosnevate) vormistamisel

kasuta osade eristamiseks suurtähti:

− akronüümid esita vaid esisuurtähega, näiteks GUI → Gui

− esita kõik sõnad suure esitähega, välja arvatud esimene

**(kaamliküüru meetod, camelBack notation)**

− esita kõik sõnad suure **algustähega (esisuurtähtede meetod,**

**InitialCaps notation)**

− eralda sõnad allkriipsuga (under\_score notation).

·

Kuigi keelereeglid otseselt ei nõua muutuja nimedele tüübi lisamist, peetakse

seda heaks stiiliks. Järgnevalt on toodud levinuimate tüüpide jaoks

kasutatavad tüüpide eesliited, koos näidetega (Hungarian notation).

**3 Funktsioonide nimede vormistamine**

Funktsioonidele nimede andmisel tuleks lähtuda verbidest ehk tegusõnadest,

ning nimed just nende hulgast valida. Nime vormistamisel kirjuta kõik nime

osad algavana suurtähega, välja arvatud esimene sõna, näiteks:

* displayImage,*

* computeBalance.*

Funktsioonide nimed võid vormistada ka eraldades erinevad nimeosad

allkriipsuga; sel juhul suurtähti ei kasutata, näiteks*:*

* display\_image,*

* compute\_balance*

**4 Konstandid**

Konstandid kirjutatakse kõik suurtähtedega. Mitmest sõnast koosneva

konstandi nime puhul kasutatakse sõnade eraldajatena allkriipsu.

Näiteks:

const double AVOGADRO\_ARV = 6.022E+23;

const float EUR\_RATE = 15.65F;

**Kokkuvõtteks võib öelda, et eksisteerib mitmeid erinevaid**

**programmeerimisstiile.** Millist ja millises mahus rakendada, sõltub juba

konkreetsest keelest ja programmeerijast . Aga miks mitte ka firmasisesest

poliitikast. Koherentses stiilis koostatud programmikoodi on lihte teistel

lugeda, seda mõista ning vajadusel ka modifitseerida. Ühe programmikoodi

piires kasuta vaid ühte kindlat stiili! Stiilireeglite eiramise ja sellest tulenevate

arusaamatuste eest vastutab ikka koodi autor ise, seega Teie!

**Koodi ülevaatuse Checklist**

Koodi üle vaatamisel peaks tähelepanu pöörama järgmistele küsimustele:

\*kas faili päises on olemas kommentaarid

o autori nimega

o faili kirjeldusega

o muutmise ajaga

o eel- ja järeltingimustega

* kas kõik meetodid (funktsioonid, protseduurid) on kommenteeritud*

* kas muutujate nimed vastavad üldlevinud kirjutusviisidele*

* kas meetodite nimed vastavad üldlevinud kirjutusviisidele*

* kas klasside nimed vastavad üldlevinud kirjutusviisidele*

* kas konstantide nimed vastavad üldlevinud kirjutusviisidele*

* kas programmikood on trepitud*

* ...*

Antud loetelu on üldine ja ei ole lõplik.

**Nõuetele vastavalt vormindatud programm**

/\*

 FAILINIMI: main.cpp

 AUTOR: Jaan Kask

 LOODUD: 20.09.2023

 MUUDETUD: 25.09.2023

 KIRJELDUS: Loob inimeste jada, prindib selle ekraanile ning samas

 prindib ka vanuste summa ja vanuste keskmine.

 Eeltingimused: ...

 Järeltingimused: ...

\*/

#include "stdafx.h"

// Inimese andmete struktuur.

struct Person

{

 char\* firstName;

 char\* familyName;

 int age;

};

/\*

 Funktsioon väljastab struktuuri Person ekraanile.

 Sisendparameetrid:

 person: struktuur Person, mis ekraanile väljastatakse.

\*/

void printPerson( Person person )

{

 // Formaadi ja väljasta ekraanile

 printf("eesnimi: %s\nperekonna nimi: %s\nvanus: %d\n",

 person.firstName,

 person.familyName,

 person.age );

}

/\*

 Arvutab inimeste vanuste summa.

Sisendparameetrid:

 person: struktuuri Person jada, millest hakatakse vanuste

 summat arvutama.

 count: struktuuri Person jada pikkus.

 Väljund: tagastab inimeste vanuste summa.

\*/

long calculateAgeSum( Person persons[], int count )

{

 long sum = 0;

 for( int i = 0 ; i < count; i++ )

 {

 sum += persons[i].age;

 }

 return sum;

}

/\*

 Arvutab inimeste vanuste keskmise.

 Sisendparameetrid:

 person: struktuuri Person jada, millest hakatakse vanuste

 keskmist arvutama.

 count: struktuuri Person jada pikkus.

 Väljund: tagastab inimeste vanuste keskmise.

\*/

float calculateAgeAverage( Person persons[], int count )

{

 float average = 0.0F;

 for( int i = 0 ; i < count; i++ )

 {

 average += persons[i].age;

 }

 return average / count;

}

/\*

 Loob inimeste jada, prindib selle ekraanile ning samas prindib ka

 vanuste summa ja vanuste keskmine.

\*/

int main( int argc, char\* argv[] )

{

 // Loo inimeste jada

 Person person[3];

 // Väärtusta jada liikmed

 person[0].firstName = "Jaan";

 person[0].familyName = "Kask";

 person[0].age = 21;

 person[1].firstName = "Mats";

 person[1].familyName = "Traat";

 person[1].age = 20;

 person[2].firstName = "Kati";

 person[2].familyName = "Karu";

 person[2].age = 20;

// Väljasta inimeste jada ekraanile

 for( int i = 0; i < 3; i++ )

 {

 printPerson( person[i] );

 }

 // Arvuta ja väljasta vanuste summa

 printf("Vanuste summa: %d\n", calculateAgeSum( person, 3 ) );

 // Arvuta ja väljasta vanuste keskmine

 printf("Vanuste keskmine: %f\n", calculateAgeAverage( person, 3 )

);

 return 0;

}