Contents

[1. Įvadas. 1](#_Toc121333568)

[2. Taisyklės kodo lygiavimui. 1](#_Toc121333569)

[2.1. Bendrosios teksto lygiavimo taisyklės. 1](#_Toc121333570)

[2.2. Tabuliacija. 2](#_Toc121333571)

[2.3. Valdymo struktūrų žymėjimas. Ciklų ir sąlyginių sakinių žymėjimas 3](#_Toc121333572)

[2.4. Funkcijų aprašai ir apibrėžimai. 3](#_Toc121333573)

[2.5. Struktūrų apibrėžimas. 4](#_Toc121333574)

[3. Taisyklės vardų parinkimui. 4](#_Toc121333575)

[3.1. Bendrosios vardų parinkimo nuostatos. 4](#_Toc121333576)

[3.2. Funkcijų vardų parinkimas. 5](#_Toc121333577)

[3.3. Kintamųjų vardų parinkimas. 5](#_Toc121333578)

[4. Taisyklės komentarams. 6](#_Toc121333579)

[4.1. Bendrosios komentarų taisyklės. 6](#_Toc121333580)

[4.2. Pradžios antraštė. 6](#_Toc121333581)

[4.3. Funkcines grupes atskiriantys komentarai. 6](#_Toc121333582)

[5. Taisyklės kodui. 7](#_Toc121333583)

# 1. Įvadas.

Hubertas Vindžigalskis. Ketvirtosios namų darbų užduoties kodavimo standartas.

# 2. Taisyklės kodo lygiavimui.

## 2.1. Bendrosios teksto lygiavimo taisyklės.

2.1.1. Tuščios eilutės yra naudojamos atskirti kintamųjų deklaravimo vietą.

2.1.2. Kintamųjų priskyrimo operatorius lygybės ženklas - „ = “ ir aritmetikos ženklai ( + , - , \*, /, tt..) iš abiejų pusių yra atskiriami tarpais.

**Tinkamas pavyzdys:** n = n - 1;

**Netinkamas pavyzdys:** n=n-1;

2.1.3. Vienam reiškiniui yra skiriama viena eilutė, valdymo struktūrai – bent 2 eilutės.

**Tinkamas pavyzdys:**

Add(head,10,10);

Print\_List(head);

int lenght = NumberOfItems(head);

**Netinkamas pavyzdys:**

Add(head,10,10); Print\_List(head); int lenght = NumberOfItems(head);

2.1.4. Bloko pradžia ir pabaiga yra lygiuojamos vertikaliai. Bloką atidarantis riestinis skliaustas yra žymimas kitoje eilutėje po kreipimosi į struktūrą / funkciją, o bloką uždarantis riestinis skliaustas visada atskirai pažymimas naujoje eilutėje.

**Tinkamai bloką apibrėžiantis pavyzdys:**

if (temp->data == data)

{

printf("The element with value %d exists in the list.\n", data);

break;

}

**Netinkamai bloką apibrėžiantis pavyzdys:**

if (temp->data == data){printf("The element with value %d exists in the list.\n", data);break;}

## 2.2. Tabuliacija.

Tabuliacija kode yra naudojama palaikyti lygių hierarchijai. Pirmąjį lygį ( kuriame nėra tabuliacijos įtraukos ) sudaro globalių kintamųjų aprašai, funkcijų ir struktūrų aprašai ir apibrėžimai, kreipimasis į preprocesoriaus direktyvas. Antrajame ( atskirtame vienu tabuliacijos žingsniu ) rašomas einamasis kodas. Atitinkamai kodas toliau yra atitraukiamas tabuliacijų pagalba, atskiriant skirtingus kodo blokų lygius, naudojantis loginiais sąryšiais.

**Tinkamas pavyzdys:**

while (temp != NULL)

{

// If the data of the current node is equal to the given data

if (temp->data == data)

{

printf("The element with value %d exists in the list.\n", data);

break;

}

// Move to the next node

temp = temp->next;

}

**Netinkamas pavyzdys:**

while (temp != NULL)

{

// If the data of the current node is equal to the given data

if (temp->data == data)

{

printf("The element with value %d exists in the list.\n", data);

break;

}

// Move to the next node

temp = temp->next;

}

## 2.3. Valdymo struktūrų žymėjimas. Ciklų ir sąlyginių sakinių žymėjimas

Lyginamuosiuose sakiniuose ir cikluose esančios loginės operacijos ( neigimas, disjunkcija, konjunkcija ir tt..) bei lyginimo operatoriai ( == , != , < , >, tt.) nuo kintamųjų iš abiejų pusių yra atskiriami tarpais.

**Tinkamas pavyzdys:**

if (temp->data == data)

**Netinkamas pavyzdys:**

if (temp->data==data)

## 2.4. Funkcijų aprašai ir apibrėžimai.

2.4.1. Funkcijos aprašui yra skiriama tik viena eilutė.

**Tinkamas funkcijos aprašas:**

void insert(int data);

**Netinkamas funcijos aprašas:**

void insert(

int data

);

2.4.2. Funkcijos bloką atidarantis riestinis skliaustas yra žymimas kitoje eilutėje po funkcijos pavadinimu, o bloką uždarantis riestinis skliaustas atskirai naujoje eilutėje po funkcijos kodo pabaigos. Funkcijos kodą sudarančios eilutės atskiriamos lygiais naudojant tabuliaciją.

**Pavyzdys:**

void insert(int data)

{

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

temp->data = data;

temp->next = head;

head = temp;

}

## 2.5. Struktūrų apibrėžimas.

Struktūros sintaksę sudaro tipo deklaravimas ir pavadinimas. Kiekvienas atskiras struktūros

elementas turi būti naujoje eilutėje. Papildomas struktūros pavadinimas deklaruojamas po uždarančių riestinių skliaustų, bet ne naujoje eilutėje.

**Tinkamas struktūros apibrėžimo pavyzdys:**

struct node {

int data;

struct node \*next;

};

**Netinkamas struktūros apibrėžimo pavyzdys:**

struct node {int data;struct node \*next;};

# 3. Taisyklės vardų parinkimui.

### 3.1. Bendrosios vardų parinkimo nuostatos.

3.1.1. Vardai yra rašomi anglų kalba.

3.1.2. Vardų pavadinimai yra parinkti prasmingai ir atitinka jų atliekamą funkciją.

### 3.2. Funkcijų vardų parinkimas.

3.2.1. Funkcijų pavadinimai pradedami mažąja raide. Jeigu funkcijos pavadinimas susidaro iš daugiau nei vieno žodžio, šis junginys yra atskiriamas kiekvieną naują žodį užrašant didžiąja raide.

**Tinkamas pavyzdys:**

void insert(int data);

**Netinkamas pavyzdys:**

void Insert(int data);

3.2.2. Funkcijų pavadinimai turi būti sudaromi iš liepiamosios nuosakos veiksmažodžių, tiesiogiai nurodančių kokį veiksmą atlieka funkcija.

**Tinkamas pavyzdys:**

void insert(int data);

**Netinkamas pavyzdys:**

void in(int data);

### 3.3. Kintamųjų vardų parinkimas.

3.3.1. Įprastų kintamųjų pavadinimai prasideda mažąja raide.

**Tinkamas pavyzdys:**

int data = 0;

**Netinkamas pavyzdys:**

Int Data = 0;

3.3.2. Jeigu norima deklaruoti kintamąjį susidarantį iš dviejų ar daugiau skirtingų žodžių, jiems atskirti naudojama didžioji raidė. **Tinkamas pavyzdys:**

int newNode; **Netinkamas pavyzdys:**

int newnode;

# 4. Taisyklės komentarams.

## 4.1. Bendrosios komentarų taisyklės.

4.1.1. Komentarų kalba sutampa su kintamųjų ir funkcijų aprašų kalba, išimtis pradžios antraštė.

4.1.2. Komentaruose negali būti tiesioginio kodo dubliavimo.

4.1.3. Komentarai negali prieštarauti kodui.

4.1.4. Komentarai yra trumpi ir lakoniški.

## 4.2. Pradžios antraštė.

Pradžios antraštė yra komentaras. Pirmoji eilutė yra sudaryta iš autoriaus vardo ir pavardės. Antroji iš studijų programos, kurso ir krypties. Trečioji nurodo namų darbų užduoties numerį. Ketvirtoje ir likusiose eilutėse yra nurodoma užduoties sąlyga.

**Tinkamas pradžios aprašo pavyzdys C kalboje:**

//Hubertas Vindzigalskis

//Programu Sistemos 1 kursas 3 grupe

//Ketvirtas namu darbas

//Sudaryti vienpusį sąrašą. Parašyti procedūrą, kuri išmeta reikšme nurodytą elementą. Jeigu tokio elemento nėra, turi būti išvestas atitinkamas pranešimas.

**Netinkamas pradžios aprašo pavyzdys C kalboje:**

//Hubertas Vindzigalskis. Programu Sistemos 1 kursas 3 grupe. Ketvirtas namu darbas. Sudaryti vienpusį sąrašą. Parašyti procedūrą, kuri išmeta reikšme nurodytą elementą. Jeigu tokio elemento nėra, turi būti išvestas atitinkamas pranešimas.

## 4.3. Funkcines grupes atskiriantys komentarai.

Funkcijų grupės atskiriamos komentarais esančiais virš funkcijų grupės, nurodančias funkcinės grupės paskirtį.

**Tinkamas pavyzdys:**

// Function to insert a node at the beginning of the list

void insert(int data)

{

...

}

// Function to search for a node with given data

void search(int data)

{

...

}

**Netinkamas pavyzdys:**

void insert(int data) // Function to insert a node at the beginning of the list

{

...

}

void search(int data) // Function to search for a node with given data

{

...

}

# 5. Taisyklės kodui.

5.1. Kodas dalinamas į funkcijas.

5.2. Kodo išdėstyme vyrauja loginė tvarka. Visų pirma įterpiami antraštės failai (.h), toliau struktūrų, o dar vėliau funkcijų aprašai. Funkcijos apibrėžimai yra deklaruojami po Main() funkcijos kodu.

5.3. Main() funkcija yra naudojama tik funkcijų realizacijai ir grįžtamojo ryšio suteikimui vartotojui.

5.4. Vengiama pasikartojančių kodo fragmentų.