**Kodavimo standartai**

*Vardai*

* Kintamieji yra aprašomi prasmingai – jie visada konkrečiai parodo savo paskirtį ir yra išreikšti trumpesniais, kartais sutrumpintais žodžiais/žodžių junginiais.

Pvz. int streetNumber; double length;

Pastaba: jeigu uždavinio sąlygoje yra apibrėžtas kintamasis, tai būtinai programoje reikės įsivesti tokio pat pavadinimo ir paskirties kintamąjį.

* Paprastų kintamųjų pavadinimai visada prasideda mažąją raide. Jeigu kintamasis yra žodžių junginys, tai kiekvienas sekantis žodis prasideda didžiąją raide.  
  Pvz. char currentSymbol;
* Rodyklės aprašomos kaip ir kiti įprasti kintamieji su keliomis išimtimis:
  + *Jeigu rodyklė rodo į masyvo pradžią, tai ji privalo prasidėti didžiąja raide (Pvz. int Array[4]).*
  + Jeigu rodyklė užfiksuotai rodo į kintamąjį, kuris yra apibrėžtas, tai jos pavadinimas bus ‘p\_<kintamojo vardas>‘ (Pvz. int \*p\_streetNumber).
* Pagal susitarimą galima apibrėžti kintamuosius su pavadinimais, kurie atitinka tam tikrą bendrąją reikšmę žinoma visiems.  
  Pvz.   
  int n; *//Nurodo elementų skaičių*  
  int i; *//Kažkuris elementas sekoje/masyve*  
  int A[7] //Masyvas (Jeigu nesugalvojame jam pavadinimo)
* ‘#define‘ tipo globalios konstantos aprašomos tik didžiosiomis raidėmis. Jeigu tai žodžių junginys, tai dedame ‘\_‘ tarp žodžių. Paprastai globalių kintamųjų geriau vengti.  
  Pvz. #define PI\_VALUE = 3.14;
* Struktūrų pavadinimai prasideda didžiąją raide, joje esantiems kintamiesiems yra taikomi įprasti standartai. Struktūros kreipinį ir vardą yra įprasta susitrumpinti su ‘typedef‘.   
  Pvz. typedef struct Node List;
* Funkcijų pavadinimai visada prasideda didžiąja raide. Jeigu programoje yra funkcijos, kurios apdoroja panašius/tuos pačius parametrus, tai jų pavadinimai gali prasidėti pirmu vienodu žodžiu, atskirtu apatiniu brūkšniu su kitais žodžiais.  
  Pvz. List\_Create(&head);  
   List\_Remove(head);

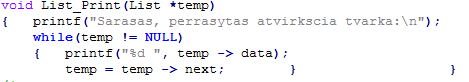
*Struktūra*

*Programos struktūra*

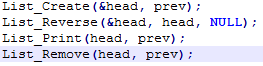
* Viršuje apibrėžiami funkcijų prototipai. Pirmoji funkcija yra main(), po jos eina kitos.
* Vengti globalių apibrėžimų ir kintamųjų.
* Tarp funkcijų yra dedamas ištisinis komentaras, kuris vizualiai padeda orientuotis tarp jų.
* Viršuje apsibrėžiamos struktūros, esant reikalui sutrumpiname jų pavadinimus su typedef.
* Stengtis kaip įmanoma per daug neišplėsti kodinių eilučių (Dėl skaitomumo).

*Skliaustų, ženklų ir aprašų pozicijos*

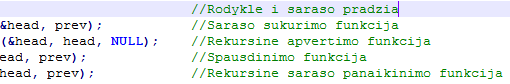
* Po kiekvienos funkcijos iškart dedamas kabliataškis.
* Funkcijos gali eiti viena po kitos toje pačioje eilutėje tik tuomet, kai jų paskirtys yra panašaus pobūdžio. Geriausia šito varianto vengti, kad kodas būtų įskaitomas. Galima apsibrėžti to paties tipo kintamuosius, atskiriant juos kableliais.
* Riestiniai skliaustai yra išdėstyti taip, kad vizualiai atvaizduoja du vienas priešais kitą esančius kampus, kurie „įrėminą“ kodo bloką. Pradžia sutampa su kodo bloko šaknine operacija (if, while, function). Pabaiga užsibaigia po paskutinės kodo eilutės – dedamas TAB tol, kol skliaustas užeis už kodo bloko ilgiausios operacijos.



Išimtis: do-while ciklo ir panašaus principo komandoms taikomi kitokios riestinių skliaustų taisyklės. Taip yra daroma, nes while(....); yra rašomas iškart po uždaromojo riestinio skliausto.

* Dedami tarpai tarp loginių ir aritmetinių operacijų ženklų. Skliaustai papildomų tarpų nereikalauja.
* Rodyklės apibrėžiamos dedant žvaigždute prie pat jos pavadinimo.   
  Pvz. int \*pointer;
* Funkcijos parametrų eiliškumas yra parenkamas autoriaus patogumui. Jeigu yra daugiau funkcijų apdorojančių tuos pačius parametrus, tai eiliškumas visur toks pat.  
  

*Komentarai*

* Komentarai dažniausiai naudojami po kintamųjų apibrėžimo. Jie detaliau paaiškina tam tikro kintamojo prasmę. Kartais ir gali būti naudojami paaiškinant tam tikrą vietą kode.
* Visi komentarai, esantys vienoje funkcijoje, gali būti naudojami tik už ilgiausios kodo eilutės ribų.  
  
* Komentarų blokai gali būti programos pradžioje/pabaigoje, kurie turi naudingos informacijos programai arba jeigu yra aprašyta jos sąlyga.