C KODO STANDARTAS

Autorius: Aringas Civilka

Pavadinimas: Asmeninis C programavimo kodo standartas

Versija: 1.0

Turinys

---

1. Kodo lygiavimo taisyklės

1.1. Indentacija

- Naudojamos **4 tarpų** įtraukos (ne tabuliacijos).

1.2. Riesti skliaustai

- Atidarantis `{` rašomas **toje pačioje eilutėje kaip ir loginio bloko pradžia**.
- Uždarantis `}` rašomas atskiroje eilutėje, lygiuojamas su atitinkamo bloko pradžia.

```
if (condition) {
    executeFunction();
}
```

- `switch` blokuose kiekvienas `case` apsupamas riestiniais skliaustais, net jei jame nėra kitų loginių blokų.

```
case 6: {
    int removed = listCompress(list->list, compareStudentId);
    printf("Removed %d elements\n", removed);
    printStudentList(list);
    break;
}
```

- Riestiniai skliaustai loginiuose blokuose praleidžiami, jeigu jų turinys susideda iš 1 eilutės ir bloko logika yra paprasta/trumpa.

```
if (index < 0 || index > listSize(list)) return 0;
```

1.3. Tarpai

- Tarpas dedamas:
  - tarp raktažodžio (while, for, if etc.) ir skliaustų (`if (x)` , ne `if(x)`),
  - aplink dvejetainius operatorius (`=` , `+` , `-` , `==`).

---

2. Vardų parinkimo taisyklės

2.1. Failų pavadinimai

- Failų pavadinimai rašomi **camelCase** stiliumi.
- Antraštiniai failai turi `.h` plėtinį, realizacijos – `.c` .

2.2. Funkcijų pavadinimai

- Funkcijų pavadinimai prasideda **veiksmažodžiu** arba atitinkamos duomenų struktūros pavadinimu.
- Naudojamas **camelCase** stilius.
- Pavadinimas aiškiai nusako funkcijos paskirtį.

```
int listInsert(List *list, int index, void *data, size_t dataSize);

void printStudentList(StudentList *slist);
```

2.3. Kintamųjų pavadinimai

- Naudojami kuo trumpesni, bet prasmę perduodantys kintamieji.
- Naudojamas **camelCase**.

2.4. Struktūrų ir tipų pavadinimai

- Struktūrų pavadinimai rašomi **PascalCase**.

```
typedef struct Student
{
    char name[50];
    int age;
} Student;

typedef int (*CompareFunc)(void *a, void *b);
```

---

3. Komentarų rašymo taisyklės

3.1. Failų antraštės komentarai

- Kiekvieno `.c` ir `.h` failo pradžioje pateikiamas komentaras, aprašantis failo paskirtį.

```
// studentList.h
// Public interface for singly linked lists that store data of type Student,
// extends the SLList.h library
```

3.2. Funkcijų komentarai

- Prieš kiekvieną viešą funkciją pateikiamas komentaras, aprašantis:
    - funkcijos paskirtį,
    - grąžinamą reikšmę (jei taikoma).

3.3. Vidiniai komentarai

- Komentarai naudojami tik ten, kur kodo logika nėra savaime aiški.
- Vengiama perteklinių komentarų, kurie kartoja patį kodą.

---

4. Bendrosios kodo rašymo taisyklės

4.1 Eilučių ilgis

- Visos eilutės turi būti iki 80 simbolių ilgio. Jei eilutė viršija šį limitą, ji suskaidoma pagal loginius taškus (&&, || etc.) arba parametrus (funkcijose).

```
int studentListInsertArray(
    StudentList *slist,
    void *array,
    size_t length,
    int index
);
```

4.2. Moduliškumas

- Kiekvienas modulis turi:
    - antraštinį (`.h`) failą – deklaracijoms,
    - realizacijos (`.c`) failą – funkcijų aprašams.

4.2. `#include` tvarka

- Pirmiausia įtraukiami vartotojo sukurti `.h` failai.
- Po to – standartinės bibliotekos.

4.4. Atminties valdymas

- Dinaminė atmintis (`malloc`) visada atlaisvinama (`free`).
- Tikrinama, ar atmintis sėkmingai paskirta.