Procedūrinio programavimo pagrindai
8 tema

Rodyklė (angl. pointers) – vienas kertinių programavimo kalbų elementų, leidžiantis klaidžioti po kompiuterio atmintį. C kalboje nuo rodyklių neatsiejamas tiek darbas su masyvais, tiek dinaminės atminties valdymas, tiek funkcijos parametrų perdavimas. Šių užduočių tikslas – susipažinti su šiais rodyklių taikymais ir išmokti jomis operuoti.
Funkcija *main* yra skirta testavimui ir demonstracijai, kaip veikia jūsų parašytos funkcijos. Nei vienoje iš žemiau užduotyse aprašomų funkcijų negali būti nei skaitymo iš, nei spausdinimo į ekraną. Specialios vartotojo sąsajos daryti nereikia – pratybų rezultatas yra funkcijos, o ne *main* esantis kodas.
Kiekvieną užduotį rašykite atskirame kodo faile. Atliktas užduotis įkelkite į VU VMA, laikydamiesi pateiktų nurodymų.

Užduotis 1a.
Apibrėžkite funkciją *createArray*, kuri dinaminėje atmintyje leidžia sukurti (naują) sveikųjų skaičių masyvą, užpildytą atsitiktinėmis reikšmėmis iš nurodyto intervalo. Ši funkcija gauna būsimo masyvo dydį *size*, išskiria atitinkamo dydžio bloką dinaminėje atmintyje, ir užpildo jį atsitiktiniais skaičiais iš intervalo [*low*; *high*]. Paskutiniajam veiksmui atlikti, jei norite, galite pasinaudoti ankstesnėse (7) pratybose parengta funkcija *generateArray*. Funkcija *createArray* sėkmės atveju turi grąžinti rodyklę į pirmą naujai sukurto masyvo elementą, nesėkmės atveju – *NULL*.
Užduotis 1b.
Įdomumo dėlei, perrašykite funkcijas *createArray* ir/ar *generateArray* taip, kad jų viduje vietoje operatoriaus [] masyvo elementui pasiekti būtų naudojamas išrodyklinimo operatorius *, adreso operatorius & ir rodyklių aritmetika. Pasikeisti turi tiktai tekstas kodo faile, o funkcijų veikimas turi išlikti nepasikeitęs.

Užduotis 2.
Apibrėžkite funkciją, kuri gauna *argc-1* failų vardų per komandinės eilutės parametrus, ir grąžina daugiausiai baitų atmintyje užimančio failo vardą.

Užduotis 3.
Apibrėžkite funkciją *swap*, kuri moka sukeisti dviejų (tai funkcijai perduodamų) kintamųjų (sveikųjų skaičių) reikšmes vietomis, taip, kad apkeitimas vyktų funkcijoje, o efektas liktų galioti ne tik funkcijoje, bet ir už jos ribų.

Užduotis 4.
Apibrėžkite funkciją *splitData*, kuri moka vieną masyvą padalinti į dvi dalis, dinaminėje atmintyje sukurdama du naujus masyvus (į pirmą masyvą įrašydama skaičius esančius pradinio masyvo pradžioje, o į antrą masyvą – pradinio masyvo likusią dalį). Ši funkcija gauna penkis parametrus. Du pirmi parametrai leidžia gauti pradinio masyvo duomenis ir dydį, trečias parametras nurodo pirmosios dalies dydį (pagal jį nustatoma, kurioje vietoje vyksta perskėlimas), o likę du parametrai skirti perduoti (grąžinti) tuos du naujai sukurtus masyvus (pirmųjų elementų adresus) iš funkcijos į likusią programą. Funkcija pirmiausia patikrina, ar visų (!) parametrų reikšmės korektiškos, tuomet sukuria du naujus atitinkamų dydžių masyvus dinaminėje atmintyje, po ko perkelia reikiamą skaičių elementų į atitinkamai pirmą ir į antrą masyvą. Funkcija tikisi, jog iki ją iškviečiant šie du masyvai dar nebus sukurti, t.y. tikisi, jog gaus parametrų reikšmes lygias *NULL* ir sukurs masyvus savo viduje. Sėkmės atveju, funkcija grąžina skaičių 0, nesėkmės atveju -1.

Papildomos užduotys.

Užduotis 5.

Apibrėžkite dvimatį masyvą, kurio elementai – kintamo ilgio tekstinės eilutės, ir apibrėžkite funkcijas, leidžiančias dirbti su šia duomenų struktūra.

a) apibrėžkite funkciją, leidžiančią sukurti dvimatį eilučių masyvą dinaminėje atmintyje, su n eilučių ir m stulpelių; visus elementus nustatykite lygius 0 (*NULL*)

b) apibrėžkite funkciją, leidžiančią įrašyti (naują) eilutę į ląstelę, esančią į-ojoje eilutėje ir j-ajame stulpelyje; funkcija gauna pozicijas ir eilutę, tada išskiria atminties bloką į kurį ir nukopijuoja perduotos eilutės turinį

c) apibrėžkite funkciją, leidžiančią gauti (grąžinti) į-ojoje eilutėje ir j-ajame stulpelyje esančios eilutės turinį

d) apibrėžkite funkciją, leidžiančią sunaikinti dvimatį masyvą kartu su visomis (dinamiškai sukurtomis) jo viduje esančiomis eilutėmis

Užduotis 6.

Apibrėžkite funkcijas *myMalloc*, *myCalloc*, *myRealloc* ir *myFree*, imituojančias C kalboje esančio dinaminės atminties skirstytojo darbą. Atmintyje (globalioje erdvėje esančiame masyve) saugokite rezervuotų atminties blokų pradžias ir jų dydžius, jiems kurti, keisti ir naikinti pasinaudokite standartinėmis dinaminio atminties valdymo funkcijomis. Papildomai apibrėžkite funkciją *getMallocSize*, leidžiančią sužinoti kiek baitų atminties rezervuota minėtomis funkcijomis; funkciją *freeAll*, leidžiančią viską atlaisvinti vienu sykiu; funkciją *defragmentMemory*, leidžiančią sudėlioti visus naudojamus (rezervuotus) atminties blokus atmintyje taip, kad tarp jų nebūtų tarpų.