BENDRASIS PROGRAMAVIMO DOKUMENTAS
EUROPOS SĄJUNGA
Europos socialinis fondas
ŠVIETIMO IR MOKSLO MINISTERIJA
VYTAUTO DIDŽIOJO UNIVERSITETAS
UNIVERSITAS VYTAUTI MAGNI
**KURKIME ATEITĮ DRAUGE!**

**2004-2006 m. Bendrojo programavimo dokumento 2 prioriteto „Žmogiškųjų išteklių plėtra“ 4 priemonė „Mokymosi visą gyvenimą sąlygų plėtra“**

Projekto sutarties numeris: **ESF/2004/2.4.0-K01-160/SUT-261**

Projekto pavadinimas: **Inovatyvūs mokymosi metodai ir naujausios technologijos gamtos mokslų bakalaurų rengimui**

---

# FIZ 313 KOMPIUTERINĖ FIZIKA

## Laboratorinis darbas

# PROCEDŪRŲ NAUDOJIMAS ŠAKOTINIAMS FIZIKOS UŽDAVINIAMS

**1. Darbo tikslas:**

1.1. Naudojant funkcijas ir paprogrames parašyti programą, kuri apskaičiuotų vandens tankį esant skirtingoms temperatūroms, t.y. skirtingose vandens fazėse.

**2. Darbo užduotis:**

2.1. Išmokti taikyti programavimą procedūromis užduočių sprendimui.

2.2. Parašyti programą, kuri apskaičiuotų duotos masės vandens tankį ir tūrį, esant atmosferos slėgiui ir pasirinktai temperatūrai. Funkcijos ir paprogramės gali būti vidinės arba išorinės.

2.3. Modifikuoti fortran projektą taip, kad funkcijos ir paprogramės būtų atskirame modulyje.

**3. Bendroji teorija**

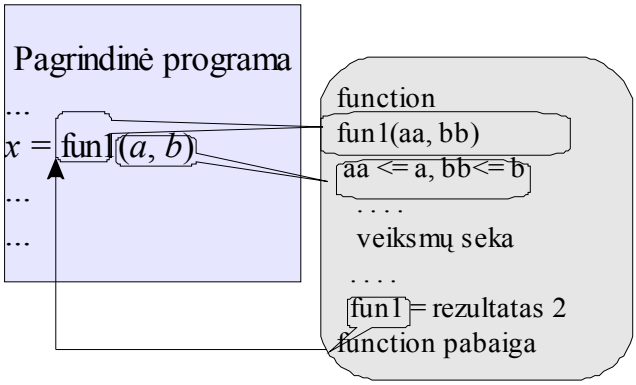
**Funkcijos struktūra FORTRANe**

Pirmiausia išsiaiškinkime, kuo skiriasi funkcija nuo paprogramės. Su standartinėmis (įdiegtosiomis) funkcijomis jau susidūrėme nagrinėdami fortran pavyzdžius. Įprastas funkcijos vartojimo pavyzdys yra toks:

```
x = fun1 (a, b, c, ...)
```

Šiame pavyzdyje funkcija yra **fun1**, o **a, b, c** vadinami funkcijos parametrai. Funkcija yra tam tikrų veiksmų seka. Kai tik parašom funkcijos vardą, tuomet programos vykdymas perduodamas į funkcijai. Kartu su valdymo perdavimu, į funkciją perduodami ir parametrų a ir b faktinės reikšmės.

Naudojant šiuos duomenis, funkcija atlieka skaičiavimus ir kintamajam x priskiria skaičiavimų rezultatą. Loginė funkcijos veikimo schema pateikta 1 paveiksle.




**1 pav. Loginė funkcijos veikimo schema.**

Funkcijoje fun1 aprašyta veiksmų seka. Parametrų aa ir bb reikšmės funkcijai nežinomos, kol į ją nesikreipiame iš pagrindinės programos, todėl funkcijos parametrai pačiame funkcijos aprašyme aa ir bb vadinami ***formaliaisiais parametrais***. Faktinės reikšmės yra a ir b, todėl funkcijos parametrai pagrindinėje programoje vadinami ***faktiškaisiais parametrais***.

Kai tik kreipiamės į funkciją, tuomet kartu su valdymo perdavimu perduodami ir faktiniai parametrai, t.y., aa = a, bb = b. Suprantama, pavadinimai aa ir bb gali būti bet kokie, svarbu, kad faktiškieji ir formalieji parametrai 1) *būtų surašyti ta pačia tvarka* ir 2) *sutaptų duomenų tipai*.

```
program funpvz
    real a, b, erz1, rez2
    a = 1.1; b = -1,6
    rez1 = sin(a)
    rez2 = cos(b)
    print *, rez1, rez2, sin(a) + cos (b)
end program
```

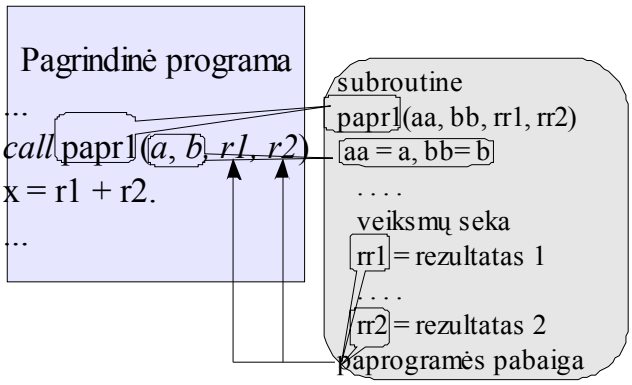
**2 pav. sin(x) ir cos(x) yra vidinės funkcijos.**

Neretai pasitaiko, kada į pagrindinę programą reikia grąžinti ne vieną rezultatą, o kelis. O funkciją perduoda tik vieną rezultatą (išskyrus tuos atvejus, kai naudojam masyvus). Norint perduoti kelis rezultatus į pagrindinę programą, naudojamas programinis vienetas, vadinamas paprograme.

Paprogramės vartojimo pavyzdys yra toks:

```
call papr1 (a, b,..., r1, r2, ...)
```

Šiame pavyzdyje paprogramės vardas papr1, faktiškieji paprogramės parametrai a ir b, o paprogramės atliktų veiksmų rezultatai bus perduoti į r1 ir **r2**. Suprantama, kad tą akimirką, kai kreipiamės į paprogramę **papr1**, tada perduodame valdymą ir faktiškuosius parametrus kaip parodyta 3 paveiksle.




**3 pav. Loginė paprogramės veikimo schema.**

Atlikus veiksmų seką paprogramėje rezultatai grąžinami į pagrindinę programą, į rezultatams skirtus kintamuosius r1 ir r2. Suprantama, ir formaliųjų, ir faktiškųjų parametrų vardai gali skirtis, tačiau jų tvarka ir tipai pagrindinėje programoje bei paprogramėje privalo sutapti.

Dabar galime apibendrinti taip:

***procedūros*** yra funkcijos ir paprogramės – tai atskiri programiniai vienetai, skirti tam tikrai užduočiai atlikti;

***paprogramė*** yra programinis vienetas, vykdantis tam tikrą užduotį ir grąžinantis į pagrindinę programą keletą rezultatų;

***funkcija*** yra supaprastintas paprogramės variantas, kai į pagrindinę programą reikia grąžinti tik vieną rezultatą.

## Vidinės procedūros

Vidinės procedūros būna išdėstytos pagrindinės programos viduje naudojant operatorių **CONTAINS**. Tokios programos struktūra pavaizduota 4 paveiksle.

```
[PROGRAM  PR vardas]
    [aprašantieji operatoriai]
    [vykdomieji operatoriai]
    CONTAINS
        VIDINĖS PROCEDŪROS
END [program [pr vardas]]
```

**4 pav. Vidinių procedūrų blokas pagrindinėje programoje formuojamas po CONTAINS.**

Žemiau parodyta pačių vidinių procedūrų konstrukcija. Vidinių paprogramių struktūra

```
SUBROUTINE PaPR_vardas(formaliųjų parametrų sąrašas)
    [aprašantieji operatoriai]
    [vykdomieji operatoriai]
END SUBROUTINE [PaPR_vardas]
```

beveik niekuo nesiskiria o vidinių funkcijų struktūros

```
FUNCTION Funkc_vardas(formaliųjų parametrų sąrašas)
    [aprašantieji operatoriai]
    [vykdomieji operatoriai]
END FUNCTION [Funkc_vardas]
```

Žemiau pateikiamas pirmiausia paprastas pavyzdys, kaip sukurti ir įtraukti į programą vidinę funkciją, o kitame pavyzdyje ta pati programa papildyta vidine paprograme.

```
program testFunkc
    real :: c1 = 6.9, c2 = 8.3
    print *,'5 + 3 = ', sudetis(5.,3.)              ! kreipiamės į funkciją,
    print *,'c1 + c2 = ', sudetis(c1,c2)            ! c1, c2 yra faktiniai
    print *,'7 - 4 = ', sudetis(7.,-4.)             ! 7 ir -4 yra faktiniai

    contains      ! "konteineryje" esanti funkcija - vidinė
    real function sudetis(aa, bb)      ! aa, bb – formalieji par.
       real aa, bb

         sudetis = aa + bb          ! sudetis  = rezultatas
    end function                    ! pereinam į kreipimosi vietą
end
```

**5 pav. Vidinės funkcijos panaudojimo pavyzdys.**

Vidinės paprogramės panaudojimo pavyzdys pateiktas žemiau. Ir funkcijoje, ir paprogramėje naudojami kintamieji (**aa, bb, ResultSud, ResultDaug**) pilnai aprašyti, pagrindinėje programoje jų aprašinėti nereikia. C1, c2, yra tikrieji parametrai, o aa, bb – formalieji; formalių parametrų reikšmės pasidaro žinomos tik tuomet, kai kreipiamės į procedūrą (1 ir 3 diagramos).

Paprogramė iškviečiama naudojant operatorių CALL. Pagrindinėje programoje rez1 ir rez2 – tai kintamieji, kuriems paprogramė perduoda atliktų skaičiavimų rezultatus, t.y., paprogramės **veiksmai** vykdymo pabaigoje atliekamas duomenų perdavimas **resultSud -> rez1, resultDaug -> rez2**. Vieną ar kitą paprogramės naudojimo schemą, t.y., duomenų perdavimo mechanizmą programuotojas turi sugalvoti pats. Kad nebūtų painiavos, kiekvienai funkcijai ir paprogramei jos autoriai paruošia išsamų aprašymą, kuriame itin kruopščiai nurodyta, ką reiškia kiekvienas kintamųjų sąrašo narys. Vėliau su tokių paprogramių pavydžiais susipažinsime.

```
program testProc
    real :: c1 = 6.9, c2 = 8.3
    real rez1, rez2
    print *,'5 + 3 = ', sudetis(5.,3.)              ! kreipiamės į funkciją
    print *,'c1 + c2 = ', sudetis(c1,c2)            ! c1, c2 faktiniai
    print *, rez1, rez2             ! patikrinam, kam lygūs rez1
    CALL veiksmai (c1, c2, rez1, rez2)              ! kreipiamės į approgr.
    print *, rez1, rez2                             !rez1 ir rez2 pasikeitė
contains
    real FUNCTION sudetis(aa, bb)
       real aa, bb
        sudetis = aa + bb
    end FUNCTION
    SUBROUTINE veiksmai(aa, bb, ResultSud,ResultDaug)
        real aa, bb                                          ! aprašantieji oper..
        real ResultSud, ResultDaug
        ResultSud = aa + bb                         ! vykdomieji oper.
```

```
        ResultDaug = aa* bb
  end SUBROUTINE                              ! į eilutę po CALL
end
```

**6 pav. Vidinių procedūrų panaudojimo pavyzdys**

## Išorinės procedūros

1) Išorinės procedūros būna išdėstytos pagrindinės programos ***išorėje*** kaip parodyta 7 pavyzdyje.

```
[PROGRAM  PR vardas]
    [aprašantieji operatoriai]
    [vykdomieji operatoriai]
END [program [pr vardas]]
    IŠORINĖS PROCEDŪROS
```

**7 pav. Išorinės procedūros yra pagrindinės programos išorėje.**

2) Kitas išorinių procedūrų skirtumas nuo vidinių procedūrų yra tai, jog jos ***gali turėti vidinių procedūrų***, t.y., išorinių paprogramių struktūra

```
SUBROUTINE PaPR_vardas(formaliųjų parametrų sąrašas)
    [aprašantieji operatoriai]
    [vykdomieji operatoriai]
    CONTAINS
        vidinės procedūros
END [SUBROUTINE [PaPR_vardas]]
```

o išorinių funkcijų tokia:

```
FUNCTION Funkc_vardas(formaliųjų parametrų sąrašas)
    [aprašantieji operatoriai]
    [vykdomieji operatoriai]
    CONTAINS
        vidinės procedūros
END [FUNCTION [Funkc_vardas]]
```

3) Be to, išorinių procedūrų aprašymuose galima išleisti baigiamąjį **subroutine** ir **function**.

4) Kadangi pagrindinė programa nežino, koks yra išorinės funkcijos grąžinamo rezultato tipas, todėl joje reikia apibrėžti funkcijos tipą.

Žemiau pateiktame programos tekste nurodyti visi išorinės funkcijos panaudojimo skirtumai nuo vidinės.

```
program sudeda1
    real sudetis                                    ! 4 PASTABA
    print *, '5 + 3 = ', sudetis(5.,3.)
    print *, '6 + 8 = ', sudetis(6.,8.)
    print *, '7 - 4 = ', sudetis(7.,-4.)
end
! ****  pagrindinės programos pabaiga    ********************
! **********                                      1 PASTABA
real function sudetis(aa, bb)
    real aa, bb                 ! aprašantieji operatoriai
    sudetis = aa + bb           ! Vykdomieji
    CONTAINS
        ! . . . . . .            2 PASTABA
end  ! [ ...]                                     ! 3 PASTABA
```

**8 pav. Išorinės funkcijos panaudojimo pavyzdys.**

Praktine svarbiausia tai, jog vidinės procedūros „nematomos“ išoriniams programiniams vienetams. Žemiau pateiktame pavyzdyje yra dvi funkcijos **sudetis** ir **daugyba**. **Sudetis** yra vidinė, o **daugyba** – išorinė funkcija. Iš funkcijos **sudetis** galima kreiptis į bet kokią išorinę funkciją, tačiau iš funkcijos **daugyba** kreipiantis į funkciją **sudetis** (į vidinę funkciją) negalima.

```
program
    real :: c1 = 6.9, c2 = 8.3
    real sudetis
    print *,'5 + 3 = ', sudetis(5.,3.)
    print *,'c1 + c2 = ', sudetis(c1,c2)
    print *,'5 * 3 = ', daugyba(5.,3.)
    print *,'c1 * c2 = ', daugyba(c1,c2)
    contains      ! "konteineryje" esanti funkcija - vidinė
    real function sudetis(aa, bb)
        real aa, bb
                sudetis = aa + bb
        print *,'5 * 4 = ', daugyba(5.,4.)          ! Kreipimasis į išorinę
     end function

end
real function daugyba(cc, dd)                       ! išorinė funkcija
    real cc, dd
    daugyba = cc * dd
    !print *,'5 + 3 = ', sudetis(5.,3.)             ! Kreipimasis į vidinę
end function
```

**9 pav. Kreipimosi į vidines ir išorines funkcijas skirtumai.**

## Moduliai

Kaip jau buvo minėta, ***modulis*** – tai savarankiškas programinis vienetas, kuriame atskirai surašytos programos dalys, kurias programuotojas praktiškai niekada nekeičia. Tai gali būti:

kintamųjų aprašymas (deklaracijos);

procedūrų sąsajos (*interface*);

modulinės procedūros;

kitos mažiau svarbios programos dalys.

Modulio struktūra yra tokia:

```
MODULE modulio_vardas
    [aprašantieji operatoriai]
    CONTAINS
        modulinės procedūros
END [MODULE [modulio_vardas]]
```

Moduliai paprastai saugomi skirtinguose failuose. Norint įtraukti modulio turinį į programinį vienetą, reikia naudoti operatorių **USE** nurodant modulio vardą. Su šiuo operatoriumi vienus modulius galima įtraukti ne tik į pagrindinę programą, bet ir į kitus modulius.

Pavyzdžiui, ankstesniuose skyreliuose aptartą aritmetinių veiksmų programą suskirstysime į du failus. Viename faile aprašysime modulį **aritmModul** (10 pav.), jo turinys pateikiamas žemiau (failą pavadinkime „aritModulis.f90“).

```
! *****************    aritModulis.f90    ************************
MODULE aritmModul
! aprašantieji operatoriai – šiame pavyzdyje jų nėra
CONTAINS
    real FUNCTION sudetis(aa, bb)
        real aa, bb
        sudetis = aa + bb
    end FUNCTION
end MODULE aritmModul
```

**10 pav. Modulio aprašyme yra tik viena funkcija „sudetis“.**

Pagrindinėje programoje (failas vardas „aritMain.f90“) modulio turinys įtraukiamas su operatoriumi USE (11 pav.) .

```
!  ****************    aritMain.f90    **************
program sudeda
    USE aritmModul

    print *, '5 + 3 = ', sudetis(5.,3.)
    print *, '6 + 8 = ', sudetis(6.,8.)
    print *, '7 - 4 = ', sudetis(7.,-4.)
end program sudeda
```

**11 pav. Pagrindinėje programoje naudojame sukurtą modulį su operatoriumi USE.**

Norint, kad abiejų failų derinys veiktų teisingai, jie abu turi būti įtraukti į tą patį projektą ir sukompliuoti atskirai. Kad FPS 4.1 aplinkoje nereiktų naršyti po kompiuterio failų sistemą, geriausia daryti taip:

1) Pirmiausia reikia sukurti failą su pagrindine programa.

2) Tda parašytą programą sukompliuoti, FPS 4.1 automatiškai sukurs naują projektą tame kataloge, kur yra failas su pagrindine programa; į kompliavimo klaidas šiame etape galima nekreipti dėmesio.3) tada sukurti failą, kuriame bus modulio turinys ir įtraukti jį į prieš tai sukurtą projektą. FPS 4.1 aplinkoje tai daroma parinkus **Insert --> Files into project**.

4) Modulio failą sukompliuoti. Įtraukti į projektą galima ir automatiškai - paleidus modulio failą kompliuoti, FPS 4.1 paklaus „**This file is not included in the project. Would you like to add it**“.

5) Tada perkompliuoti failą su pagrindine programa. Dabar kompliatorius projekto kataloge ras objektinį failą „aritModul.obj“ su visa informacija, reikalinga pagrindiniam failui kompliuoti bei vykdyti.

## 4. Tyrimo metodika

1. Vandens fazinę būseną šiame darbe nustatoma pagal temperatūrą: jei $0^oC < T < 100\ ^oC$, vanduo yra skystojoje būsenoje, todėl jo tankis 1000 kg/m$^3$. Jei vandens temperatūra $T < 0^oC$, vanduo kietojoje būsenoje, todėl jo tankis 900 kg/m$^3$. Jei vandens temperatūra $T > 100^oC$, jis yra dujinėje būsenoje, tokio vandens tankį apskaičiuojame pagal Mendelejevo-Klapeirono lygtį:

$$pV = \frac{m}{\mu}RT$$

kur $p$ – dujų slėgis (šiuo atveju vandens garų slėgis lygus 1 atmosferai, t.y., p = 101 kPa), $V$ – kūno užimamas tūris, $m$ – kūno masė, $R$ – idealiųjų dujų konstanta, lygi , T – temperatūra, kuri bendru atveju gali būti bet kokia reali (200 – 500 K).

## 5. Tyrimo eiga

1. Laboratorinis darbas atliekamas kompiuterių klasėje.

2. Išanalizuokite užduotį, nubraižykite algoritmo eskizą, pažymėkite dydžius, kuriuos reikia suskaičiuoti, taip pat numatykite faktinius bei formaliuosius parametrus.

3. Parašykite visas reikalingas procedūras ir sukurkite programą reikalingų dydžių apskaičiavimui pagal užduotyje pateikiamus reikalavimus.

4. Programa turi užklausti, kokia vandens masė ir kokia temperatūra, o tada apskaičiuoti šio vandens tūrį ir tankį. Į ekraną turi būti išvedami: a) aplinkos sąlygos (slėgis, temperatūra), b) vandens medžiaginiai parametrai (tūris, tankis), c) visi dydžiai turi būti pateikti su vienetais.

5. Atlikti užduotį dviem būdais: funkcijas ir paprogrames aprašant a) tame pačiame fortran programos faile ir b) aprašant atskirame modulyje.

6. Rašant programą laikytis tvarkingo rašymo taisyklių, t.y., konstrukcijų blokai turi matytis pagal „reljefinį“ kodo užrašymą.

7. Atsiskaitomų darbų projektai turi būti saugomi tvarkingoje asmeninių failų struktūroje. Atsiskaitymo metu mokėti paaiškinti su fortran programos kompliavimu ir vykdymu susijusius klausimus FPS 4.1 sistemoje.

## 6. Kontroliniai kiausimai

1. Loginė programos struktūra, procedūrų reikalingumo pagrįstumas.
2. Funkcijų bendrosios savybės ir naudojimo ypatybės.
3. Paprogramių bendrosios savybės ir naudojimo ypatybės.

4. Procedūrų formalieji ir faktiniai parametrai.
5. Pateiktosios programos struktūra, veikimo principas.
6. Mokėti paaiškinti: a) kur yra formalieji procedūrų parametrai, o kur faktiniai; b) kuriame programos taške vyksta parametrų perdavimas į procedūrą ir kuriame taške – sugrąžinamos procedūrose apskaičiuotų kintamųjų reikšmės; c) paaiškinti, kuriame programos vykdymo taške procedūrų kintamieji įgyja reikšmes.

## 7. Literatūra


1. A. Kanapickas. Paskaitų konspektas. 4 skyrius.

2. MS FPS 4.1 pagalbos sistema: *Microsoft Developer Studio User's Guide -> Chapter 8: Subprograms and Modules.*

3. R. Belevičius, R. Kutas. FORTRANAS. Vadovėlis. V.: 1998. 241 p., 7 skyrius (paprogramiai) ir 8 skyrius (Naujos FORTRAN galimybės).