Procedūrinio programavimo pagrindai

Išvestiniai duomenų tipai

lekt. Irmantas Radavičius
irmantas.radavicius@mif.vu.lt
Informatikos institutas, MIF, VU

Turinys

Tipų konvertavimas

Automatinis (angl. implicit) konvertavimas

- skirtingų tipų operandai
- skirtingų tipų funkcijų parametrai

Išreikštinis (angl. explicit) konvertavimas (operatoriumi)

- (tipas) reiškinys

  (int)1.0

  (signed char)65

Tipų konvertavimas (priskyrimas)

Iš “siauresnio” tipo į “platesnį”

```
int i = (int)‘A’;
float f = (float)(1.0 + 1);
```

Iš “platesnio” tipo į “siauresnį”

```
char c = (char)1000;
int x = (int)1.0;
```

Simboliai

Gali būti naudojami kaip skaitinės reikšmės

char = signed char? char = unsigned char?

Standartinių simbolių reikšmės – teigiamos

Po konversijos į int – teigiamas? neigiamas?

Skaičiai

"Siauresnis" tipas konvertuojamas į "platesnį"

Rezultatas – "platesnio" tipo

Supaprastintos taisyklės:

If either operand is `long double`, convert the other to `long double`.

Otherwise, if either operand is `double`, convert the other to `double`.

Otherwise, if either operand is `float`, convert the other to `float`.

Otherwise, convert `char` and `short` to `int`.

Then, if either operand is `long`, convert the other to `long`.

Skaičiai (signed/unsigned)

Verčiant į unsigned, skaičiuojama “moduliu n”

Verčiant į signed, reikšmė išlieka, arba (jei ne) priklauso nuo realizacijos

First, if either operand is `long double`, the other is converted to `long double`.
Otherwise, if either operand is `double`, the other is converted to `double`.
Otherwise, if either operand is `float`, the other is converted to `float`.
Otherwise, the integral promotions are performed on both operands; then, if either operand is `unsigned long int`, the other is converted to `unsigned long int`.
Otherwise, if one operand is `long int` and the other is `unsigned int`, the effect depends on whether a `long int` can represent all values of an `unsigned int`; if so, the `unsigned int` operand is converted to `long int`; if not, both are converted to `unsigned long int`.
Otherwise, if one operand is `long int`, the other is converted to `long int`.
Otherwise, if either operand is `unsigned int`, the other is converted to `unsigned int`.
Otherwise, both operands have type `int`.

Realūs skaičiai

Konvertuojant į int, trupmeninė dalis atmetama

Konvertuojant iš double į float, atmetama? apvalinama?

Jei reikšmė netelpa, visais atvejais rezultatas neprognozuojamas

Išvestiniai duomenų tipai

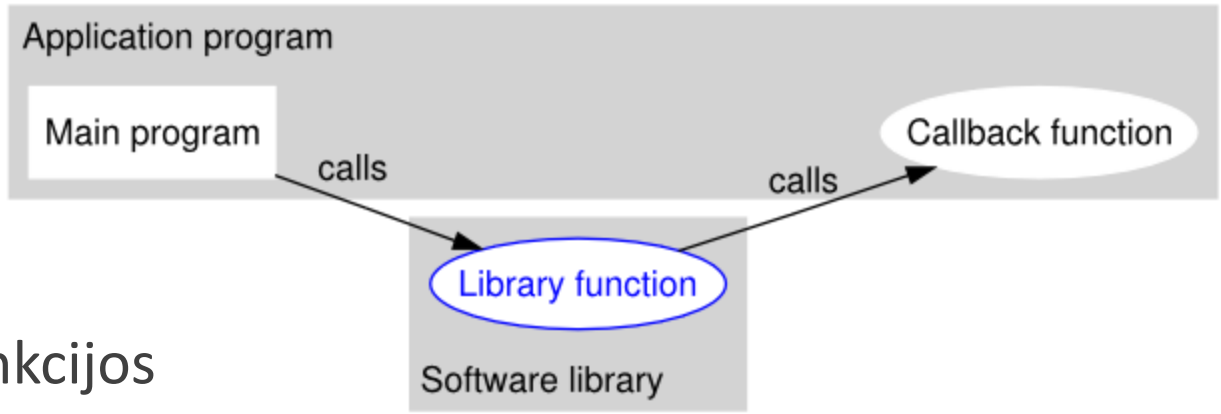
- rodyklės
- masyvai
- funkcijos
- struktūros
- sąjungos
- enumeratoriai

- rodyklių masyvai
- rodyklės į masyvus
- rodyklės į funkcijas
- ...

Rodyklės į funkcijas

```
int f(int);        // funkcija su parametru int, grąžinanti int
int * x(int);      // funkcija su parametru int, grąžinanti rodyklę į int
int (*y)(int);     // rodyklė į funkciją su parametru int, grąžinančią int

y = &f;            // alternatyva y = f;
(*y)(1);           // alternatyva  y(1);
```



Panaudojimas:

atgalinio iškvietimo funkcijos

Rodyklės į funkcijas

main.c

```c
#include <stdio.h>
#include <stdlib.h>

char *msg = "Hello\n";



void print(char *str){
    printf("%s", str);
}

int main (){
    print(msg);
    return 0;
}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>

char *msg(void){
    return "Hello\n";
};

void print(char *(*str)(void)){
    printf("%s", str());
}

int main (){
    print(msg);
    return 0;
}
```

Komplikuoti aprašai

```
int a;
int *d;
int (*g)();
int *j[2];
int(*m[2])();
int p(int (*)());
```

```
int b();
int *e();
int (*h)(int);
int (*k)[2];
int*(*n)(int*[2]);
int(*r())(int);
```

```
int c(int);
int *f(int *);
int *(*i)(int *);
int *(*l)[2];
int(*(*o)[2])();
int (*(*q)(int (*)()))();
```

OPERATORS	ASSOCIATIVITY
`()  []  ->  .`	left to right
`!  ~  ++  --  +  -  *  &  (type)  sizeof`	right to left

Struktūrų apibrėžimas

Struktūros naudojamos skirtingų (nebūtinai) tipų duomenims grupuoti.

Struktūrinio duomenų tipo apibrėžimas:

```
struct Vardas {
        tipas narys1, narys2, …;
        tipas narys3;
        …
};
```

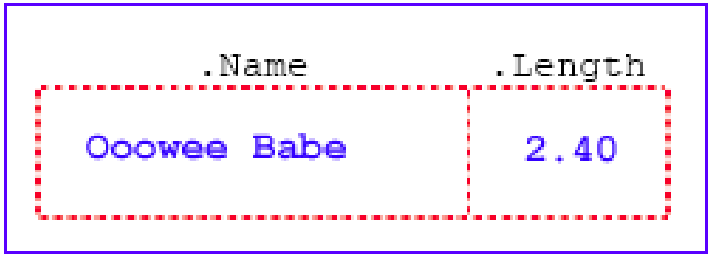
Struktūros (kintamojo) apibrėžimas:

```
struct Vardas vardas;
```

Struktūrų panaudojimas

Tipo apibrėžimas, kintamojo apibrėžimas, kintamojo inicializavimas

```
struct song {
        char *name;
        float length;
} song = { “Ooowee Babe”, 2.4 };
```



Svarbu: “song” nėra tipo vardas (C), “struct song” – yra

Operacijos: priskyrimas, adreso paėmimas, kreipimasis į narius

```
struct song s1 = song, *s2 = &song;
s1.length = 2.4; s2->length= 2.4; (*s2).length= 2.4; // prioritetai!
```

Struktūrų specifika

```
int a1[10], a2[10]; a1 = a2;                          // draudžiama
struct s { char c; int val; } s1, s2; s1 = s2;        // leidžiama

struct s data[100]; data[0].val = 1;                  // struktūrų masyvas
struct s *p = (struct s *)malloc(5); ++p;             // struktūros dydis, 5??

struct ss { struct s s; int a1[20]; } ss1;            // įdėtinės struktūros
struct { struct s s; int a1[20]; } ss2;               // anoniminės struktūros
ss1 = ss2; ss1 = (struct ss)ss2;                      // draudžiama

struct xx someFunc (struct xx *someStruct);           // funkcija, perdavimas
```

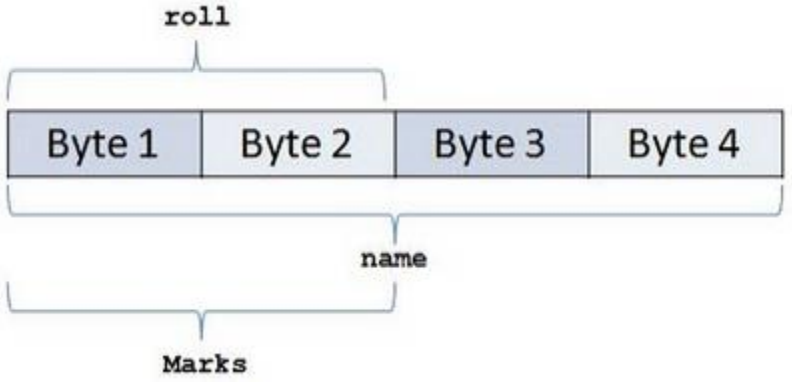
Sąjungos

struct s { laukas1; laukas2; … }; // laukai turi savo vietą atmintyje

union u { laukas1; laukas2; … }; // laukai dalijasi vietą atmintyje

```
union stud
{
int roll;
char name[4];
int marks;
}s1;
```

Member	Memory Required
Roll	2
Name	4
Marks	2



Visa kita –
analogiškai struktūroms.
Panaudojimas:
taupyti, stebėti atmintį

Enumeratoriai

Alternatyva sveikojo tipo konstantoms apibrėžti (C)

enum Vardas { REIKŠMĖ1 = 0, REIKŠMĖ2 = 1, … } kintamasis;

```
enum escapes { BELL = '\a', BACKSPACE = '\b', TAB = '\t',
               NEWLINE = '\n', VTAB = '\v', RETURN = '\r' };

enum months { JAN = 1, FEB, MAR, APR, MAY, JUN,
              JUL, AUG, SEP, OCT, NOV, DEC };
                    /* FEB is 2, MAR is 3, etc. */
```

Savybės:

galima inicializuoti sveikosiomis reikšmėmis

pagal nutylėjimą pradedama nuliu ir didinama vienetu

reikšmės gali sutapti, vardai turi skirtis

Typedef

Leidžia žinomiems duomenų tipams suteikti papildomus vardus.

typedef old_type New_type;

typedef int Index;
typedef char *String;
typedef struct MyStruct { ... } MyStruct;

Panaudojimas:
paprastumui, modifikuojamumui užtikrinti, dokumentavimui

Duomenų struktūros

“Daugiamačiai” duomenų tipai (pavyzdžiai?)

- masyvai masyvuose
- struktūros masyvuose
- masyvai struktūrose
- struktūros struktūrose

```
struct x { struct y yValue; } xValue;        // gerai
struct x { struct x xValue; } xValue;        // blogai
struct x { struct x *xPtr; } xValue;         // gerai
```

Dinaminių duomenų struktūrų realizacija

Dinaminiai masyvai

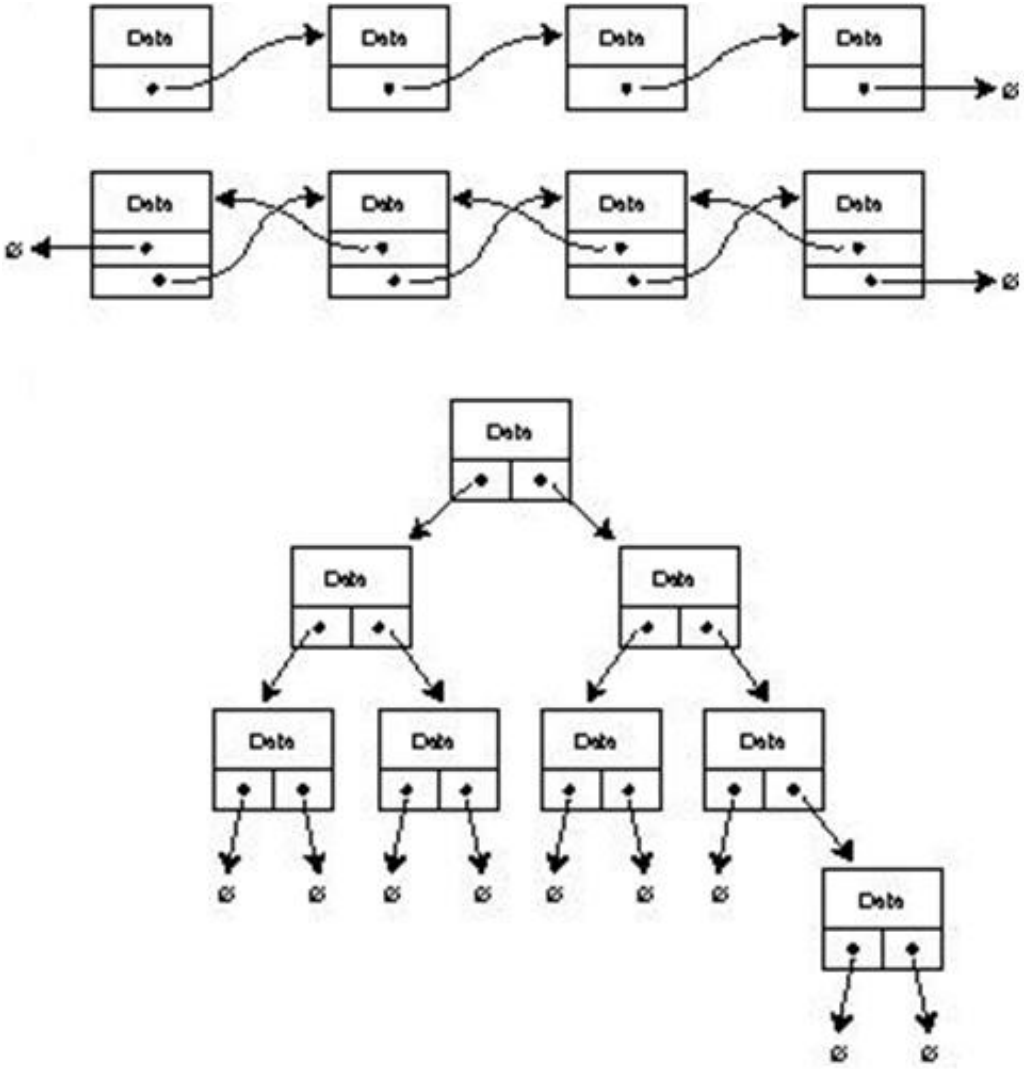
Dinaminiai sąrašai

- vienpusiai
- dvipusiai
- cikliniai

Medžiai

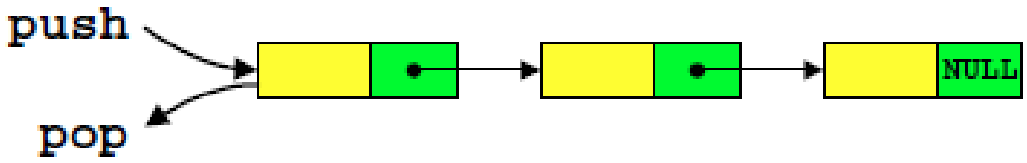
- dvejetainiai
- kiti

...

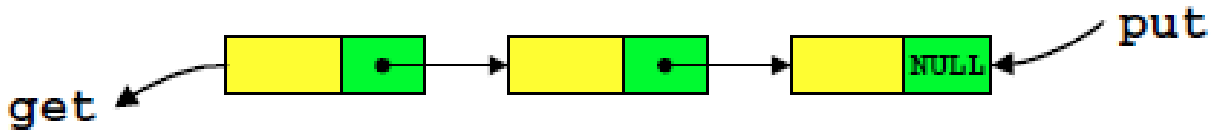


Dinaminių duomenų struktūrų logika

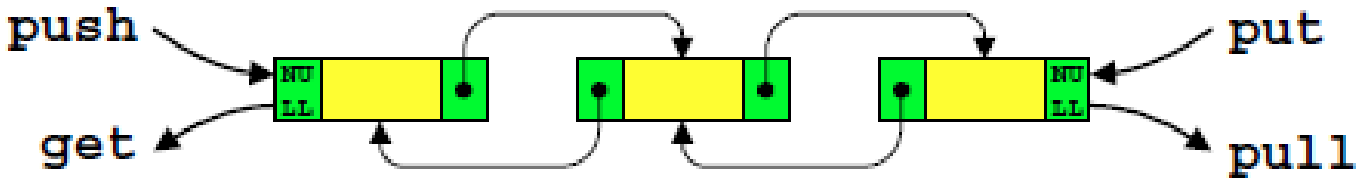
Stekas



Eilė



Dekas



...

```
struct { time_t ends; } lecture = { NOW };
```