Pagrindiniai duomenų tipai

Konspektas

Baziniai duomenų tipai

Tik C++

Kiekvienam duomenų tipui apibrėžiama, kokios gali būti jo reikšmės ir kokius apdorojimo veiksmus šioms reikšmėms leidžiama taikyti. Kiekvienoje programavimo kalboje yra grupė pagrindinių duomenų struktūrų (tipų), kurias realizuoja tų kalbų kompiliatoriai. Naudojant šias struktūras, projektuojamos konkretiems uždaviniams pritaikytos išvestinės struktūros. Norint programoje naudoti pagrindinių tipų duomenis, pakanka aprašyti juos realizuojančius kintamuosius ir žinoti šiems tipams taikomus veiksmus. Pagrindiniai C++ kalbos duomenų tipai skiriami sveikiesiems skaičiams, realiesiems skaičiams, kompiuterio alfabeto simboliams ir loginėms reikšmėms registruoti.

1 lentelė. Sveikojo tipo atmainos, kai mašininio žodžio ilgis 32 bitai

Žymėjimas	Reikšmių atkarpa	Lietuviškas pavadinimas	Baitų sk.
short int arba short	-32768 ÷ 32767	Trumpas sveikasis skaičius	2
int	$-2^{31} \div (2^{31}-1)$	Sveikasis skaičius	4
long int arba long	$-2^{31} \div (2^{31}-1)$	Ilgas sveikasis skaičius	4
long long	$-2^{63} \div (2^{63}-1)$	Dvigubas ilgasis skaičius	8

Duomenų tipo užimamos atmintinės dydis, rašymo į atmintinę, skaitymo ir apdorojimo sparta priklauso nuo jo kodavimo būdų. Todėl, siekiant, kad būtų galima geriau išnaudoti kompiuterių įrangą, realizuojama po keletą kiekvieno pagrindinio tipo atmainų. Sveikųjų ir realiųjų tipų atmainos aprašytos atitinkamai 1 ir 2 lentelėse.

Konkrečios sveikojo tipo atmainos savybės ir jo užimamas atmintinės lauko dydis priklauso nuo naudojamo kompiliatoriaus ir architektūros, tačiau visada garantuojama, kad tipui short skiriama ne mažiau kaip 16 bitų (5 dešimtainiai skaitmenys), o tipui long - ne mažiau kaip 32 bitų (10 dešimtainių skaitmenų). Tipui int skiriamo lauko dydis paprastai sutampa su kompiuteryje duomenų mainams naudojamų mašininių žodžių ilgiu. Todėl jis gali sutapti su short tipu (16 bitų procesoriai) arba su long tipu (32 bitų procesoriai). Šiuolaikiniuose asmeniniuose kompiuteriuose int tipui dažniausiai skiriami 4 baitai. Naujuose kompiliatoriuose taip pat galima naudoti dvigubą ilgąjį skaičių (long long).

Papildant sveikųjų skaičių tipų vardus požymiu unsigned, sukuriamos jų modifikacijos be ženklo, kurios negali turėti neigiamų reikšmių, o didžiausia teigiama reikšmė gali būti du kartus didesnė:

unsigned short - trumpas sveikasis skaičius be ženklo,

unsigned int — sveikasis skaičius be ženklo,

unsigned long - ilgas sveikasis skaičius be ženklo.

1.6.2 lentelė. Realiojo tipo atmainos

Žymėjimas	Reikšmių atkarpa	Lietuviškas pavadinimas	Baitų sk.
float	-3.4E+38 ÷ 3.4E+38	Realusis skaičius	4
double	-1.6E+308 ÷ .7E+308	Dvigubo tikslumo skaičius	8
long double	-1.6E+308 ÷1.6E+308	Ilgas realusis skaičius	10

Realiojo tipo atmainos skiriasi leistinų reikšmių atkarpomis ir reikšmių vaizdavimo tikslumu. Atmainos float reikšmės registruojamos 6 dešimtainių ženklų tikslumu, double - 15 ženklų tikslumu, o long double — 20 ženklų tikslumu.

Simbolinis tipas žymimas vardu char, o jo reikšmės yra kompiuterio naudojamo alfabeto kodai. Šis tipas suderinamas su sveikuoju tipu, todėl su simbolių kodais galima atlikti visus sveikiesiems skaičiams taikomus veiksmus.

Programos tikrinamoms sąlygoms aprašyti naudojamas loginis tipas, kuris gali įgauti tiktai dvi reikšmes - true (tiesa) arba falše (melas). Ilgą laiką C++ kalboje loginio tipo nebuvo, nes loginės reikšmės koduojamos sveikaisiais skaičiais: reikšmė falše koduojama skaičiumi 0, o visi kiti sveikieji skaičiai interpretuojami kaip reikšmė true. Šiuo metu ANSII C++ standarte loginis tipas yra numatytas. Jis žymimas vardu bool, reikšmė false koduojama skaičiumi 0, o true — skaičiumi 1. Tokiam kodavimui užtektų vieno dvejetainio simbolio (bito), tačiau kompiuteriuose mažiausias adresuojamų duomenų vienetas yra baitas, todėl loginėms reikšmėms koduoti naudojami vieno baito dydžio kodai.

Duomenims saugoti programose sukuriami kintamieji, aprašomi tokios sandaros sakiniais (aprašais):

Tipo vardas Kintamųjų sąrašas;

Jei sakinys aprašo grupę kintamųjų, iš jų vardų sudaromas sąrašas, kurio elementai atskiriami kableliais. Kintamųjų aprašai ir kitų tipų sakiniai vienas nuo kito atskiriami kabliataškiais. Šioje knygoje, aprašant sakinių ir kitų C++ kalbos struktūrų rašymo taisykles (sintaksę), kursyvu žymimi paaiškinimai, nurodantys, kas atitinkamoje struktūros vietoje turi būti rašoma.

Kintamųjų aprašų pavyzdžiai:

int suma, n, kl, First_Class; float x; double y;

Kintamųjų reikšmės programose formuojamos specialiais duomenų įvedimo iš kompiuterio išorinių įrenginių veiksmais arba priskyrimo operatoriais, kurių aprašymo sintaksė tokia:

Kintamojo vardas = Reikšmė;

Simboliu = žymimo priskyrimo operatoriaus prasmė yra „suteikti reikšmę". Kairėje operatoriaus pusėje nurodomas kintamojo, kuriam skiriama reikšmė, vardas, o dešinėje — šią reikšmę arba jos skaičiavimą aprašanti išraiška. Kai ta pati reikšmė suteikiama grupei kintamųjų, naudojama sintaksinė struktūra:

Kintamojo vardas = Kintamojo vardas . . . = Kintamojo vardas = Reikšmė;

Paprasčiausios konstantų aprašymo išraiškos. Konstantos būna tokių pat tipų kaip ir kintamieji, bet jos dažniausiai nurodomos ne vardais, o reikšmėmis. Šios reikšmės gali būti papildomos specialiais simboliais, nurodančiais, kaip jas reikia interpretuoti. Pavyzdžiui, sveikojo tipo konstantas galima rašyti dešimtainėje, aštuntainėje ir šešioliktainėje skaičiavimo sistemose. Be jokių papildomų žymėjimų tradiciniu būdu užrašyti sveikieji skaičiai laikomi dešimtainiais:

10, 1234, -27, 123456789012345 Aštuntainiai skaičiai pažymimi jų pradžioje rašant nulį:

0, 02, 077, 0123 Šešioliktainių skaičių pradžioje rašoma simbolių pora 0x;

0x0, 0x2, 0x3fa, 0x53DA

Simboliai a, b, c, d, e, f arba analogiškos didžiosios raidės žymi šešioliktainius skaitmenis — atitinkamai 10, 11, 12, 13, 14 ir 15. Šešioliktainiai ir aštuntainiai skaičiai plačiai naudojami užrašant adresus ir dvejetainių vektorių reikšmes.

Sveikosiose konstantose priesagomis U ir L galima nurodyti jų atmainas: U — unsigned (be ženklo) ir L — long (ilgas):

10U, 123456789012345L

Realiuosius skaičius leidžiama rašyti tiek tradicine forma, atskiriant sveikąją dalį nuo trupmeninės dešimtainiu tašku, tiek rodykline forma. Pastaruoju atveju naudojamas simbolis E arba e, kurio prasmė yra „dešimt pakelta laipsniu". Priesagomis f (float) ir 1 (long double) nurodomos realiųjų skaičių atmainos. Kai atmaina nenurodoma, kompiliatorius parenka tipinę atmainą double.

Realiojo tipo konstantų pavyzdžiai:

-0.125, 12.3E12, .15e-6f, 15.7el251;

Simbolių reikšmės rašomos tarp apostrofų, pavyzdžiui: '5', 'a', 'A', 'Ž'. Siūloma atkreipti dėmesį į tai, kad šios reikšmės gali būti ne tik lotyniški C++ kalbos, bet ir kiti kompiuterio alfabeto simboliai, kuriuos galima įvesti klaviatūra bei stebėti ekrane. Jeigu jūsų kompiuteris turi lietuvišką klaviatūros tvarkyklę, tokie simboliai gali būti ir lietuviškos raidės.

Kompiuteriais tvarkant simbolinius duomenis, plačiai naudojami specialūs ryšio kanalų ir duomenų atvaizdavimo įrenginių valdymo simboliai. C++ kalboje jie turi standartinius vardus, kurie prasideda simboliu \. Dažniausiai naudojami šie valdantieji simboliai:

nauja eilutė (LF) - \n, horizontalus tabuliavimas (HT) - \t, vertikalus tabuliavimas (VT) - \v, trynimo simbolis Backspace (BS) - \b, grįžimo į eilutės pradžią simbolis (CR) - \r, perėjimas į naują eilutę (FF) - \f, simbolis \ - \\, apostrofo simbolis 0 - Y, dviguba kabutė (") - \",

Simbolius taip pat leidžiama nurodyti jų kodais. Tada naudojama tokia struktūra:

'\Kodas'

Simbolių kodus, kurie gali būti nurodomi tiek dešimtainėmis, tiek aštuntainėmis, tiek šešioliktainėmis konstantomis, reglamentuoja ASCII standartas. Simbolių aprašymo pavyzdžiai:

'\n', '\\\ ,'\t\ 'V", '\97\ '\067\ '\x5f\ 'A', 'd1.

Kintamųjų aprašuose priskyrimo operatoriais leidžiama nurodyti pradines jų reikšmes:

```
int n = 0;

double x = 12.3E12, s = 0;

char c = 'n', tab = '\t';
```

***

Kiekvienas kintamasis, kurį norima naudoti C++ kalboje turi būti priskirtas kažkuriam tipui. Priskirtas tipas kompiliatoriui nurodo, kiek vietos atmintyje reikės išskirti kintamajam, kokios operacijos ir kokie rezultatų vaizdavimo būdai bus leidžiami. Vadinasi neteisingai priskirtas tipas neišvengiamai lems netesingą programos vykdymą. Pagrindiniai tipai pateikti 1 lentelėje. Joje pateiktos ne tik baziniai C++ naudojami kintamųjų tipai, bei ir Borland Delfi aplinkoje sutinkami analogai.

Delphi	Reikšmė/dydis	C++
ShortInt	8-bit integer	signed char
SmallInt	16-bit integer	short
LongInt	32-bit integer	int
Byte	8-bit unsigned integer	unsigned char
Word	16-bit unsigned integer	unsigned short
Integer	32-bit integer	int
Cardinal	32-bit unsigned integer	unsigned int
Boolean	true/false	bool
ByteBool	true/false or 8-bit unsigned integer	unsigned char
WordBool	true/false or 16-bit unsigned integer	unsigned short
AnsiChar	8-bit unsigned character	char
Char	8-bit unsigned character	char
Single	32-bit floating point number	float
Double	64-bit floating point number	double
Extended	80-bit floating point number	long double
Real	32-bit floating point number	double

Bitų kiekis nurodo maksimalų įrašomą skaičių. Pagal jį galima surasti ir maksimalią reikšmę – jei bitų yra N, tuomet maksimalus galimas įrašyti skaičius yra 2^N-1:

N=8; 2^8-1=255.

Jei naudojamas kintamųjų tipas yra su ženklu, tuomet į maksimalus skaičius dalinasi pusiau (vienas dalis į neigiamą pusę, kita į teigiamą)

Standartinės aritmetinės operacijos C++ kalboje

Skaičiavimo procedūros C++ kalboje pačiu paprasčiausiu atveju yra standartinės kaip ir kitose kalbose – pradžioje rašomas kintamasis, kuriam priskiriamas skaičiavimo rezultatas, vėliau pati skaičiavimo operacija su joje dalyvaujančiais kintamaisiais:

atsakymas = kintamasis1 <operacija> kintamasis2

Dažniausiai naudojamos operacijos:

+ sudėties veiksmas ir teigiamas skaičiaus ženklas;

- atimties veiksmas ir neigiamas skaičiaus ženklas;

* daugyba;

/ dalyba;

% tik sveikajam tipui taikomas liekanos skaičiavimo veiksmas.

Skaičiavimo išraiškose svarbi aprašomų veiksmų vykdymo tvarka, kurią valdo operatorių prioritetai, asociatyvumo savybės ir lenktiniai skliaustai. Aukščiausią prioritetą turi operacijos, parašytos skliaustuose. Toliau seka daugyba, dalyba ir sumos bei skirtumo operacijos.

C++ kaaboje galioja sutrumpintos operacijų formos:

+= **=** ***=** **/=**

Užrašas y+=x; atitinka y=y+x;

Užrašas y*=x; atitinka y=y*x;

Dar paprasčiau realizuojamas mažinimas arba didinimas vienetu:

```
y=x++; //priskyrimas                ir                didinimas              vienetu
y=++x; //didinimas               vienetu                   ir               priskyrimas
y=--x; //mazinimas               vienetu                   ir               priskyrimas
y=x--; //priskyrimas ir mazinimas vienetu
```

Svarbu atsiminti, jog operacijos rezultato tipas nustatomas ne pagal rezultato, o pagal argumentų tipus. Todėl jei atliekama operacija su dviem sveikais skaičiais, tai ir rezultatas bus sveikas skaičius, nepriklausomai nuo to, į kokį kintamąjį jis įrašomas. Pvz. atlikus tokią skaičiavimo operaciją:

```
float                                                                    sk;
sk=2/3;
```

atsakymas bus 0, tuo tarpu tūrėtų būti 0,666. Taip yra todėl, jog atliekant dalybą iš pradžių kompiuterio atmintyje gaunamas rezultatas 0 (sveikoji dalis), o tik paskui ji įrašoma į sk kintamąjį. Norint gauti realų sveikųjų skaičių dalybos rezultatą, išraiškai reikia taikyti tipų keitimo (*type casting*) veiksmą, kuris aprašomas tokia sintaksės struktūra:

(<naujas tipo vardas>)<pertvarkoma išraiška>

Tuomet pvz. atrodytų taip:

Pastaba: skirtinguose kompiliatoriuose duomenų tipai gali būti realizuoti

skirtingai, t.y. gali skirtis šioje lentelėje nurodyti reikšmių diapazonai, užimamos

atminties dydis.

Nuorodos

http://www.mif.vu.lt/~valund/KONSPEKTAI/INFORMATIKA%20C++.pdf

http://youtu.be/Lu--kEPQ2uw

Priedas

Baziniai C ir C++ tipai

Rūšis	*Tipas*	*C89* *C99*	*C++*	*Galimi ekvivalentai*
Loginis	`bool`		✓	
	`_Bool`	✓		
Tuščias	`void`	✓	✓	

Simboliniai	**char**	✓	✓	signed char *arba* unsigned char
	signed char	✓	✓	
	unsigned char	✓	✓	
	wchar_t		✓	
Sveikieji	short	✓	✓	short int signed short signed short int
	unsigned short	✓	✓	unsigned short int
	int	✓	✓	signed signed int
	unsigned	✓	✓	unsigned int
	long	✓	✓	long int signed long signed long int
	unsigned long	✓	✓	unsigned long int
	long long	✓		long long int signed long long signed long long int

	`unsigned long long`	✓		`unsigned long long int`
Realūs	**`float`**	✓	✓	
	`double`	✓	✓	
	`long double`	✓	✓	
	`_Complex`	✓		
	`_Imaginary`	✓		

C ir C++ operacijos

Nr.	Operacija	C	C++	Prasmė
1	`::`		✓	Globalaus konteksto
	`::`		✓	Klasės arba vardų erdvės konteksto
2	`()`	✓	✓	Kreipinio į funkciją
	`[]`	✓	✓	Masyvo indekso
	`.` `->`	✓	✓	Struktūros (klasės) nario priėjimo turint struktūros kintamąjį arba rodyklę į struktūrą
	`++` `--`	✓	✓	Aritmetinės : postfiksinė inkremento, postfiksinė dekremento
	6 C++ operacijos		✓	Tipo konvertavimo ir nustatymo
3	`+` `-`	✓	✓	Aritmetinės : unarinio pliuso ir minuso
	`++` `--`	✓	✓	Aritmetinės : prefiksinė inkremento, prefiksinė dekremento
	`&`	✓	✓	Adreso
	`*`	✓	✓	Rodyklės
	`!`	✓	✓	Loginė : neiginio
	`~`	✓	✓	Bitinė : neiginio
	`sizeof`	✓	✓	Tipo (reiškinio tipo) dydžio (baitais)
	`(`tipas`)`	✓	✓	Tipo konvertavimo
	`new` `delete`		✓	Dinaminės atminties paskyrimo, panaikinimo
4	`.*` `->*`		✓	Klasės nario priėjimo per rodyklę į klasės narį, turint klasės kintamąjį (objektą) arba rodyklę į klasę

5	`*` `/` `%`	✓	✓	Aritmetinės : daugybos, dalybos, modulio
6	`+` `-`	✓	✓	Aritmetinės : sumos, atimties
7	`>>` `<<`	✓	✓	Bitinės : postūmio į dešinę, į kairę
8	`<` `<=` `>` `=>`	✓	✓	Santykio: mažiau, mažiau arba lygu, daugiau, daugiau arba lygu
9	`==` `!=`	✓	✓	Lygybės : lygu, nelygu
10	`&`	✓	✓	Bitinė : IR
11	`^`	✓	✓	Bitinė : griežtas ARBA
12	`|`	✓	✓	Bitinė : ARBA
13	`&&`	✓	✓	Loginė : IR
14	`||`	✓	✓	Loginė : ARBA
15	`?:`	✓	✓	Sąlygos
16	`=` `+=` `-=` `*=` `/=` `%=` `>>=` `<<=` `&=` `^=` `|=`	✓	✓	Priskyrimo : paprastoji, sudėtinės
17	`throw`		✓	Išskirtinių situacijų apdorojimo
18	`,`	✓	✓	Kablelio