

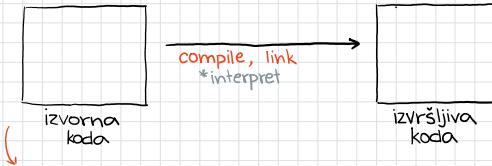
C++ sintaksa

petek, 18. oktober 2024 16:50

```
#include <iostream>           ← header file library (input, output)
using namespace std;         ← standardna imena

int main() {                 ← funkcija (izvede se koda znotraj {} )
    cout << "Hello World!";   ← objekt cout in operator <<
    return 0;                ← konec funkcije
}
```

Prevajanje - pretvorba kode v obliko, razumljivo računalniku



PREVAJANJE (compiling):

- prevorba kode v obliko, razumljivo računalniku (izrajni jezik)
- Many kot je programski jezik podoben strojnemu, bolj kompleksen je compiler
- najprej se program prevede, šele nato izvaja
- med prevajanjem se odkrijejo sintaktične napake

Interpretiranje* - vrstica po vrstica; sprotno prevajanje in nato izvajanje kode (prevajanje in izvajanje se prepletata)

↳ pocasno

Prevajalniki in interpretatorji so tudi programi - hitrost, učinkovitost, ne zelo obsežni

→ PROCES PREVAJANJA

1. urejanje (editing)
 2. predprocesiranje (preprocessing)
 3. prevajanje (compiling)
 4. povezovanje (linking)
 5. nalaganje (loading)
 6. izvajanje (executing)
- * UNIX / GNU, prevajalnik g++
 izvorna dat.
 g++ -Wall naloga.cxx -o naloga
 izpis opozoril
 na sekundarni pomnilnik (npr. disk)
- izvorni program (source code), shranimo kot datoteko .cpp ali .cxx
 - urejanje (urejalniki pogosto vgrajeni)
 - predprocesorske direktive (#include, #define...)
 - prevorba v objektno kodo (tekstova dat. v objekt file) le priskrbimo izvorno kodo compilerju
 - ↳ compiler mora vedeti, kako naj obravnava neko datoteko
 - ↳ "translation unit" # "cpp file" (nevredno → npr. več .cpp datotek znotraj ene, nastal bo le en .obj file)
 - povezovanje (linking) - C++ potrebuje reference na funkc., definirane kje drugje (knjižnice)
 - ↳ "luknje" zaradi teh manjkočih delov
 - ↳ linker poveže obj. kodo s kodo manjkočih funkc.
 - nalaganje - namestitev programa v pomnilnik (kopiranje iz diska v pomn.)
 - izvedba (nadzor CPU)

Pregled jezika C++

- case-sensitive
- characters in tables ASCII * črke Števke posebni
- vsebuje te vrste besed:
 - ✓ ključne besede - rezervirane (do, if, float...)
 - ✓ identifikatorji - poimenovanje tipov, spr., funkc...)
 - ✓ literali - zapisemo neposredno v program (cela in realna št, konsl., nizi)
 - ✓ operatrorji - izražanje operacij
 - ✓ ločila - ločevanje delov programa

Literali

- a. celo števila: desetiški, osmiški, šestnajstiški zapis
 ↳ 0, 0X1 - NAPAČNO
 ↳ short - brez pripone, long - pripona L ali ↳ 2.3e1 - NAPAČNO
- b. realna števila: decimalne, potence
 ↳ double - brez pripone, long double - pripona l ali L, float - pripona f ali F
- c. znakovne konstante: en znak, med '' tudi ubežno zaporedje*
 ↳ npr. 'a', 'r', '\n', '\x26', ''
- d. nizi: zaporedja znakov, med "", dolžina niza = št. znakov

Razpoznavanje besed: besede pišemo sk., ko to ni dvoumno npr. a= i++5 - ali je i++ ali i++?

v C++ se razpozna najdaljša možna beseda.

Komentarji: compiler jih prezre, se NE gnezdi

- /* */ - lahko v več vrsticah
- // - vrstični

↑ #

Predprocesorske direktive: pri prevajjanju se izvršijo prve

- lahko se pojavijo kjerkoli v dat. in od tam veljajo do konca
- obdelava teksta v izvornih dat. (vključevanje dat., izpustitev delov...)

To so npr. #define #elif #endif #include #else...

OUTPUT:

- Besedilo: cout - "character out"
 - za novo vrstico potrebujemo: \n ali endl
- ↳ ... presledek \... backslash \... narekovaj
- Stevila:
 - ↳ lahko s cout, npr. cout << 3+3 → 6

NAPAKE:

- a. Sintaktične: "pravopisne" - npr. raba vejc, oklepajev
- compiler jih zazna in prijavi (program ne bo deloval)

int x = 5 ← manjka podprtje

- b. Semantične: "pomenške" - npr. raba napačnih operatorev
- sintaktična pravilnost, a napačno delovanje

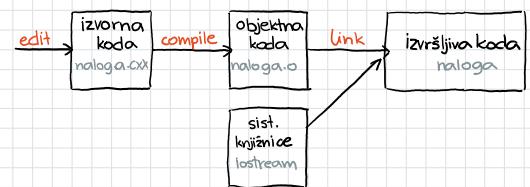
int a = 5;
 int b = 2;
 float result = a/b; [deljenje celih št. vme celo št. in
 šele nato pretvori v float]

int sum; [sum ni inicializirana (nedoločena vrednost)]
 sum += 10;

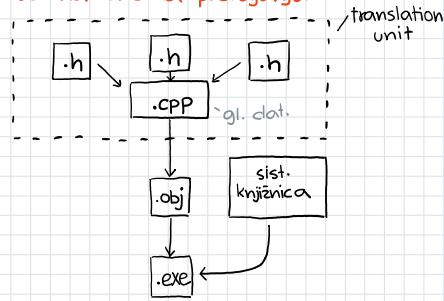
int x = 1234567890; ← prevelika vrednost za tip int

- c. Logične: problem v pristopu, postopku

→ Shema:



Še ena shema prevajanja:



→ operatrorji v C++

- ..
- [] () →
- ++ -- : ~ &
- * / % + -
- << >>
- == !=
- & ^ |
- && ||
- ? :
- = *= /= %= += -= >>= << &= ^= |=

Deklaracije, definicije:

- povezovanje identifikatorja s tipom, spremenljivko, funkcijo...
- deklaracija vpije identifikator
 - definicija poda ostale potrebe podatke

Podatkovni tipi

torek, 05. november 2024 17:51

Pregled:

int	- integer, za celo števila (+ in -) 4 byte (-2mrd.- 2mrd.)
float	- decimalke ("single precision") 4 byte
double	- decimalke ("double precision") 8 byte
char	- character, za posamezne značke 1 byte
string	- besedilo	
bool	- Boolean, true / false 1 byte

→ extended int :

short	- mojhna št. 2 byte
long] dolga št. 4 ali 8 byte
long long	

unsigned int - brez predznaka ... 4 byte

1. Celoštevilčni: short, int, long signed unsigned

tip	zlogi	signed		unsigned	
		min	max	min	max
short	2	-32768	32767	0	65535
int	4	-2147483648	2147483637	0	4294967295
long	4	-2147483648	2147483637	0	4294967295

2. Znakovni char

- 1 bit
- Signed : -128 → 127
- Unsigned : 0 → 255

ASCII *

3. Naštveni enum

- sami določimo ime in vrsto tipa, ki jih lahko zavzame
- večja berljivost in zanesljivost

enum dan {pon, tor, sre, cet, pet, sob, ned};

int main () {

 dan danes = sre;

 // dan jutri = danes + 1; // NAPAKA

 dan jutri = (int) danes + 1;

 cout << "Jutri je dan številka " << jutri << endl;

→ Jutri je dan številka 3.

4. Realni float, double, long double

e ← eksponent
m · r
mantisa ← baza (običajno št. 2)

floating-point representation ← samo m in e
- samo Q št. [-R, R]
- natančnost odvisna od bitov mantise

5. Logični bool

enum bool {false, true}
○ ↗ vse ostalo

Operatorji in izrazi

torek, 05. november 2024 18:34

Izraz → operandi - nad njimi se operacija izvrši
→ operatorji - določijo vrsto operacije

Operacije → prioriteta - katera op. imata prednost
asociativnost - leva → desna ali obratno
↳ ce ima več op. isto prioritetno

Izrazi se pri izvajanjju ovrednotijo (dajo vrednost)
→ poznamo več vrst:

1. Prireditveni (= * = += &= ...)

- preprosto ali večkratno prirejanje
- vrčanje vrednosti
- $i = i + 20$ (ali $i += 20$)

2. Relacijski (== != < > <= >=)

- primerjanje dveh vrednosti glede na relacijo, ki jo določi operator

3. Aritmetični (+ * / %)

- deljenje: $3/2 = 3.0/2.0 = 1.5 !$
- za ostanek pri delj. (%) potrebujemo cela št.

4. Pogojni (pogoj ? izraz_1 : izraz_2)

↳ če pogoj drži

5. Logični (|| &&)

- ↳ logični in in ali
- ↳ glej izjave

6. Bitni (| ^ &)

- ↳ primerjava bitov
- ↳ eks. in ink. ali in in operandov
- ↳ oboj operandi celi št.

INKLUZIVNI ALI		EKSKLUZIVNI ALI	
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

→ sestavljanje vrednosti

00010000 [] 16(00)
00100110 38(00)
00110110 54(00)

→ testiranje (zamima nas, ali je test. bit 0 ali 1)

- maska za bit + spr.
- vsi razen test. bit so enaki 0

STANDARDNE PRETVORBE:

mešani izrazi - različni tipi operandov
↳ compiler jih skuša poenotiti (če mu ne uspe - error)

char, short, enum → int

un. char, un. short → un. int

→ če je izraz še vedno mešan:

int < un. int < long < un. long < float < double < long double

Primer: char c; int i; unsigned u;

double d; long l; short s;

c - s / i → int u * 3 - i → un. int
u * 3.0 - i → double! f * 3 - i → float!
c + 1 → int 3 * s * 1 → long int
c + 1.01 → long double d + s → double

7. Pomikalni (<< >>)

- vsak operator zahteva 2 celost. tipa
- Pomik levo: prazni desni biti se zap. z 0
- Pomik desno: enako, ce pomikamo bite v unassigned st. - v assigned se vsi premaknje desno, le bit za predenjak se ohrani

8. Pretvorbeni (pretvorbeni operator)

- pretvorba izraza v tip, zapisan v ()
- Standardne pretvorbe: mešani izrazi, prevajalnik pretvori sam

9. Prefiksni (++ -- + - - ! _)

- povečamo/zmanjšamo za 1
- nad spr. se izvede operacija, ta spr. pa se nato uporabi v izrazu, kjer nastopa

10. Postfiksni (++ -- ...)

- enako kot pri (9) se spremeni vrednost za 1
- AMPAK: znatnoj izraza se najprej ovrednoti izraz s privorno spr. in se šele potem inkrementira / dekrementira

11. Konstantni

- stavek case in zač. vrednost našt. tipa,
- ↳ izraze s sizeof....
- ↳ izrazi, kjer lahko ovrednotimo že v času prevajanja

Primer:

int vsota = 0, stevilo = 10;
vsota = ++stevilo

cout << vsota << endl; ← 11
cout << stevilo << endl; ← 11

int vsota = 0, stevilo = 10;
vsota = stevilo++

cout << vsota << endl; ← 10
cout << stevilo << endl; ← 11

↑ stevilo se najprej uporabi v privredni obli, zato je vsota enaka 10

Drugi: literali, klici funkcij, dostop do konst. in spr.

* paz na prioriteto operatorjev!

Krmilne strukture

četrtek, 07. november 2024 12:09

- strukturirano programiranje

Zaporedje ("sequence") - stavki se izvajajo drug za drugim, kot so zapisani

Izbira („selection“) - če želimo izvedbo stavka, ki ni v zaporedju (izberemo določen stavek)

Ponavljanje („iteration“) - izvajanje nekaterih stakov večkrat (zanke oz. loops)

IF STAVEK:

Preprost: if (izraz) { ← če izraz drži, se stavek izvrši,
 stavek;
 } sicer se preskoči.

Kompleksnejši: if (izraz) {
 stavek 1,
 else
 stavek 2;
}

Primer: #include <iostream>
using namespace std;

```
int main () {
    int stevilo_tock;
    cout << "Vnesite število točk: ";
    cin << stevilo_tock;
    if (stevilo_tock >= 50) {
        cout << "Čestitam! Opravili ste izpit." << endl;
    } else
        cout << "Več sreče prihodnjic." << endl;
    return 0;
}
```

```
int main () {
    int nLeto;
    cout << "Vnesite letnico svojega rojstva:";
    cin >> nLeto;

    if (nLeto > 2000) {
        cout << "Rojeni ste v 21. stoletju." << endl;
    }

    else (x > y) {
        cout << "Rojeni ste ";
        if (nLeto < 1950) {
            Cout << "v prvi polovici";
        }

        else {
            cout << "v drugi polovici";
        }
        cout << " 20. stoletja." << endl;
    }

    return 0;
}
```

Osnovna struktura

- glava : pogoji za izvršitev strukture
(izbiro ali ponavljanje)
 - telo : določa stavek, nad katerim se kmilna struktura izvaja

SWITCH, CASE in DEFAULT

Osnovne oblike

switch (izraz) > case konst.izraz: > default:
stavek stavek stavek

Znotraj SWITCH imamo navadno več CASE stavkov.

1. Najprej se ovrednoti izraz.
 2. Krmiljenje se prenese v tisti case stavek, katerega konst. izraz se ujema z ovrednoteno vred. izraza.
 3. Če se NE ujema z nobenim konst. izrazom, se izvaja default (če ga ni, se switch samo zaključi).