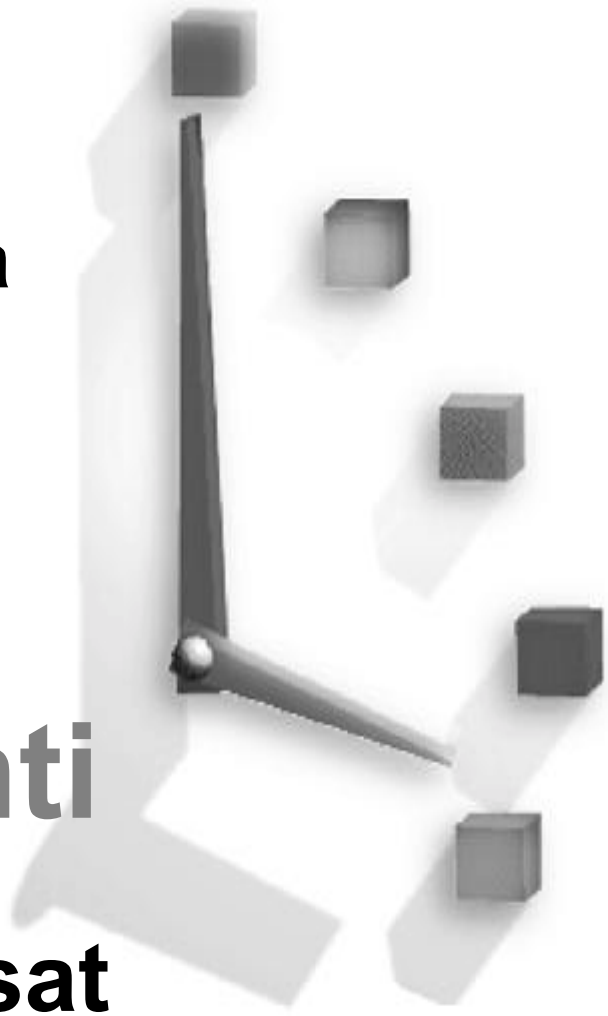
Osa

I



2. oppitunti

C++-ohjelman osat

Ennen kuin menemme yksityiskohtaisemmin sisälle C++-luokkiin, -muuttujiin jne, katsokaamme ensin, millaisista osista C++-ohjelma koostuu. Tämän tunnin aikana opit seuraavat asiat:

- C++-ohjelman osat
- Osien yhteistyö
- Mikä on funktio ja mitä se tekee

Yksinkertaisen ohjelman osat

Ensimmäisen luvun esimerkkiohjelmassa, HELLO.CPP, oli monia kiinnostavia osia. Tässä jaksossa tutkimme ohjelmaa yksityiskohtaisemmin. Ohjelma on kirjoitettu uudelleen listaukseen 2.1.

Listaus 2.1. HELLO.CPP esittelee C++ -ohjelman osat.

```
14: #include <iostream.h>
15:
16: int main()
17: {
```

```
18:    cout << "Hello World!\n";
19:    return 0;
20: }
```

Tulostus
Hello World!

Analyysi
Rivillä 1 sisällytetään ohjelmaan kirjastotiedosto IOSTREAM.H. Kääntäjän kannalta tiedosto on ikään kuin kirjoitettu suoraan lähdekoodiin juuri ohjelman alkuun.

#include-komennon tutkiminen merkki merkiltä

Uusi käsite	Ensimmäinen merkki, risuaitamerkki #, on signaali esikäsittelijälle. Mikä on sitten tuo esikäsittelijä?

Aina, kun ajat kääntäjän, se kutsuu ensin toista ohjelmaa eli esikäsittelijää. Esikäsittelijää ei tarvitse kutsua itse suoraan, vaan sitä kutsutaan automaattisesti joka kerran kääntäjää ajettaessa.

Esikäsittelijän tehtävänä on käydä läpi koodi rivi riviltä ja etsiä ne rivit, jotka alkavat #-merkillä. Kun esikäsittelijä havaitsee tuollaisen rivin, se alkaa muokata koodia. Muokattu koodi luovutetaan sitten kääntäjälle.

Komento include on esikäsittelijän komento, joka sanoo "Seuraavana on tiedoston nimi. Hae tuo tiedosto ja lue se tähän". Kulmasulkumerkit tiedoston nimen molemmin puolin kehottavat esikäsittelijää "hakemaan tiedostoa kaikista yleisistä paikoista". Jos kääntäjän asetukset ovat oikein, esikäsittelijä hakee tiedostoa hakemistosta, jossa säilytetään kaikkia .H-tiedostoja. Tiedosto Iostream.h (input/output stream) on cout-olion käytössä silloin, kun se tulostaa tekstiä näytölle.

Rivin 1 ansiosta tiedosto iostream.h sijoitetaan koodiin aivan kuin olisit kirjoittanut sen itse sinne. Kääntäjä näkee sitten koodin paikallaan jatkaessaan kääntämistä.

Analyysi rivi riviltä

Rivi 3 aloittaa todellisen ohjelman funktiolla main(). Jokaisessa C++ -ohjelmassa on main()-funktio. Funktio on koodilohko, joka suorittaa jonkun toiminnon. Funktiot toteuttavat tehtävänsä toisten funktioiden kutsumina, mutta main()-funktio on ainutlaatuinen. Kun ohjelma käynnistetään, kutsutaan main()-funktiota automaattisesti.

Kaikkien funktioiden, niin myös main()-funktion, on määritettävä, millaisen arvon se palauttaa. Main()-funktio on siitä erikoinen, että se palauttaa aina

int-tyyppisen arvon eli siis kokonaislukuarvon. Funktion arvon palauttamista käsitellään tarkemmin luvussa 4, "Ilmaukset ja ohjelmalauseet".

Kaikki funktiot alkavat aloittavalla aaltosululla ({) ja päättyvät lopettavaan aaltosulkuun (}). Main()-funktion aaltosulut ovat riveillä 4 ja 7. Kaikki aaltosulkujen väliin jäävät rivit kuuluvat funktion runkoon.

Ohjelman pääsisältö on rivillä 5. Siinä käytetään cout-oliota tulostamaan viesti näytölle. Oliota käsitellään tarkemmin luvussa 7, "Lisää luokista". Cout- ja cin-oliot tulevat kääntäjän mukana ja niiden avulla voidaan hoitaa tulostusvirtoja: tulostaa näytölle ja lukea näppäimistöltä.

Cout-oliota käytetään seuraavasti: Kirjoitetaan ensin avainsana cout ja sen jälkeen tulostuksen uudelleenohjausoperaattori (<<). Sen jälkeen kirjoitetaan tulostettava teksti. Jos haluat tulostaa merkkijonon, sijoita se lainausmerkkeihin (") kuten rivillä 5.

Uusi käsite Merkkijono on sarja tulostettavia merkkejä.

Kaksi viimeistä merkkiä, \n, kertovat cout-oliolle, että sen on sijoitettava uusi rivi sanojen Hello World! jälkeen.

Main()-funktio päättyy rivin 7 lopettavaan aaltosulkuun.

Kommentit

Ohjelmaa kirjoitettaessa kaikki koodi saattaa tuntua selvältä. Kuitenkin, kun katsot koodia uudelleen muutaman viikon tai kuukauden jälkeen, se ei enää näytäkään niin ymmärrettävältä.

Jotta koodi olisi luettavampaa ja selkeämpää, on hyvä lisätä koodiin selventäviä kommentteja. Kommentit ovat tekstiä, jonka kääntäjä ohittaa. Niissä on kuitenkin informaatiota koodin lukijalle.

Eri tyyppiset kommentit

Uusi käsite C++ -kielessä on kahden tyyppisiä kommentteja. Kommentti, joka alkaa kahdella kauttaviivalla (//) kertoo kääntäjälle, että sen tulee ohittaa kaikki teksti rivin loppuun saakka. Tämä kommentointitapa on C++ -tyylin mukainen.

Kauttaviiva ja asteriski (/*) kertoo kääntäjälle, että sen tulee ohittaa kaikki teksti aina seuraavaan asteriski ja kauttaviiva (*/) merkkiyhdistelmään saakka. Tämä kommentointitapa periytyy C++ -kieleen C-kielestä. Muista, että jokaista alkavaa /*-yhdistelmää tulee vastata päättävä */-yhdistelmä.

Useat C++ -ohjelmoijat käyttävät C++ -tyylistä kommentointia kaiken aikaa ja C-tyylistä kommentointia vain pitkien selitystekstien yhteydessä tai ohittaessaan useita koodirivejä testausvaiheessa. Voit sijoittaa C++ -tyylisiä kommentteja C-tyylisen kommentin sisään. Kuitenkin kaikki, myös C++ -tyyliset kommentit, ohitetaan niiden ollessa C-tyylisen kommentin sisällä.

Kääntäjä ohittaa siis kommentit eivätkä ne vaikuta suorituskykyyn. Listaus 2.2 havainnollistaa tätä.

Listaus 2.2. HELLO.CPP esittelee kommentteja.

```
1:  #include <iostream.h>
2:
3:  int main()
4:  {
5:    /* tämä on kommentti
6:    ja se jatkuu, kunnes kohdataan
7:    lopettava kommenttimerkki */
8:    cout << "Hello World!\n";
9:    // tämä kommentti on yhdellä rivillä
10:   cout << "That comment ended!";
11:
12:   // tuplakauttaviiva-kommentti on vain yhdellä rivillä
13:   /* asteriski-kauttaviiva -kommentit monella rivillä */
14:   return 0;
15: }
```

Tulostus

```
Hello World!
That comment ended!
```

Analyysi

Rivien 5 ja 7 välissä olevat kommentit ohitetaan kääntäjän toimesta samoin kuin riveillä 9, 12 ja 13 olevat kommentit. Rivin 9 kommentti päättyi rivin lopussa, kun taas rivien 5 ja 13 kommentit vaativat lopettavan kommenttimerkin.

Kommenttien käyttäminen

Vain harva ohjelmoija osaa kirjoittaa hyvät kommentit. Jotkut hosujat eivät kirjoita kommentteja lainkaan, mutta he eivät pysy kovinkaan kauan minun alaisinani.

Jotkut kommentoivat jokaisen rivin, mikä ei juurikaan lisää koodin arvoa, mutta tekee siitä helposti vaikealukuisen.

Kuinka kommentteja tulisi sitten käyttää? Tässä on eräs peukalosääntö: Oleta, että yleisösi osaa lukea C++ -kieltä, mutta ei sinun ajatuksiasi. Anna lähdekoodin kertoa, mitä olet tekemässä ja käytä kommentteja selittämään, miksi olet tekemässä sitä.

Tee/Älä tee

Lisää kommentteja koodiisi.
Pidä kommentit ajan tasalla.
Käytä kommentteja kertomaan, mitä koodijakso tekee.
Älä käytä kommentteja itsestään selvässä koodissa.

Funktiot

Main() on epätavallinen funktio. Käyttöjärjestelmä kutsuu main()-funktiota käynnistämään ohjelman. Main()-funktio tai jotkut toiset funktiot voivat taasen kutsua muita funktioita ohjelman ajon aikana.

Main()-funktio palauttaa aina int-arvon. Tulet näkemään seuraavissa luvuissa, että muut funktiot voivat palauttaa muun tyyppisiä arvoja tai mitään arvoa.

Ohjelma suoritetaan rivi riviltä siinä järjestyksessä kuin se esiintyy koodissa, kunnes tapahtuu funktion kutsuminen. Tällöin ohjelma haarautuu suorittamaan funktion. Kun funktio päättyy, se palauttaa ohjauksen funktion kutsua seuraavalle riville.

Kuvittele, että olet piirtämässä kuvaa itsestäsi. Piirrät pään, silmät, nenän, mutta yhtä äkkiä kynäsi terä menee poikki. Haaraudut nyt teroittamaan kynää. Tällöin keskeytät piirtämisen, nouset ylös, haet teroittimen ja teroitat kynän. Sen jälkeen palaat jatkamaan siitä, mihin aiemmin jäit.

Kun ohjelman on suoritettava jokin erikoistoiminto, se kutsuu funktiota ja funktion päättymisen jälkeen ohjelma jatkaa taas eteenpäin. Listaus 2.3 esittelee tätä ajatusta.

Listaus 2.3. Funktion kutsumisen esittely.

```
1:  #include <iostream.h>
2:
3:  // funktio DemonstrationFunction()
4:  // tulostaahyödyllisen viestin
5:  void DemonstrationFunction()
6:  {
7:    cout << "In Demonstration Function\n";
8:  }
9:
10: // main tulostaa viestin ja
11: // kutsuu funktiota DemonstrationFunction, ja
12: // tulosta toisen viestin.
13: int main()
```

```
14: {
15:    cout << "In main\n" ;
16:    DemonstrationFunction();
17:    cout << "Back in main\n";
18:    return 0;
19: }
```

Tulostus

In main
In Demonstration Function
Back in main

Analyysi

Riveillä 3-5 esitellään funktio DemonstrationFunction(). Kun sitä kutsutaan, se tulostaa viestin näytölle ja päättyy.

Rivillä 13 on todellisen ohjelman alku. Rivillä 15 tulostaa main()-funktio viestin näytölle. Viestin kirjoittamisen jälkeen kutsutaan rivillä 16 DemonstrationFunction-funktiota. Tällöin suoritetaan kyseisen funktion sisällä olevat komennot. Tässä tapauksessa funktion koodi on rivillä 7, jossa tulostetaan viesti näytölle. Funktio päättyy rivillä 8, jolloin ohjaus palaa kutsun jälkeiselle riville. Tässä tapauksessa ohjaus palautuu riville 17, jossa main() tulostaa viimeisen viestin.

Funktioiden käyttö

Funktiot palauttavat joko arvon tai ei lainkaan arvoa eli voidin. Funktio, joka summaa kaksi kokonaislukua, palauttaa summan eli kokonaislukuarvon. Funktio, joka pelkästään tulostaa viestin, ei ehkä palauta mitään arvoa, jolloin funktion sanotaan palauttavan void-arvon.

Funktiot koostuvat otsikosta ja rungosta. Otsikko puolestaan sisältää palautettavan tyypin, funktion nimen ja parametrit. Parametrien avulla voidaan funktiolle viedä arvoja. Niinpä, jos funktio summaa kaksi kokonaislukua, olisivat nuo luvut funktion parametreina. Seuraavassa on esimerkki tyypillisestä funktiosta:

```
int Sum(int a, int b)
```

Parametri kertoo, minkä tyyppisiä arvoja voidaan funktiolle viedä. Noita todellisia arvoja kutsutaan argumenteiksi. Usein käsitteitä parametri ja argumentti pidetään synonyymeinä, mutta ne ovat kuitenkin eri käsitteitä. Tässä kirjassa käsitteitä ei eroteta toisistaan.

Uusi käsite Funktion nimeä ja parametreja (ilman palautusarvoa) kutsutaan funktion allekirjoitukseksi.

Funktion runko alkaa aloittavasta aaltosulusta, jonka jälkeen on yksi tai useampi komentolause ja lopussa sitten päättävä aaltosulku. Ohjelmalauseet suorittavat funktion tehtävän. Funktio saattaa palauttaa

arvon, jolloin siinä on return-lause. Tämä lause myös keskeyttää funktion. Jos funktiossa ei ole return-lausetta, se palauttaa automaattisesti void-arvon. Palautettavan arvon täytyy olla samaa tyyppiä kuin funktion otsikossa.

Listaus 2.4 esittelee funktion, joka ottaa kaksi kokonaislukua parametreikseen ja palauttaa kokonaislukuarvon. Käsittelemme kokonaislukuarvoja ja niiden operaatioita myöhemmin tässä kirjassa.

Listaus 2.4. FUNC.CPP esittelee yksinkertaisen funktion.

```
1:  #include <iostream.h>
2:  int Add (int x, int y)
3:  {
4:
5:    cout << "In Add(), received " << x << " and " << y << "\n";
6:    return (x+y);
7:  }
8:
9:  int main()
10: {
11:   cout << "I'm in main()!\n";
12:   int a, b, c;
13:   cout << "Enter two numbers: ";
14:   cin >> a;
15:   cin >> b;
16:   cout << "\nCalling Add()\n";
17:   c=Add(a,b);
18:   cout << "\nBack in main().\n";
19:   cout << "c was set to " << c;
20:   cout << "\nExiting...\n\n";
21:   return 0;
22: }
```

Tulostus

```
I'm in main()
Enter two numbers: 3 5
Calling Add()
In Add(), received 3 and 5

Back in main()
C was set to 8
Exiting…
```

Analyysi

Funktio Add() esitellään rivillä 2. Se ottaa kaksi kokonaislukuparametria ja palauttaa kokonaislukuarvon. Itse ohjelma alkaa rivillä 11, jossa se tulostaa viestin. Ohjelma pyytää antamaan kaksi kokonaislukua (rivit 13-15). Käyttäjä antaa numerot erotettuina välilyönnillä ja painaa Enter-näppäintä. Main() vie nuo luvut Add()-funktiolle rivillä 17.

Ohjelman suoritus haarautuu nyt Add()-funktioon, joka alkaa riviltä 2. Parametrit a ja b tulostetaan ja lasketaan sitten yhteen. Tulos palautetaan rivillä 6 ja funktio päättyy.

Riveillä 14 ja 15 käytetään cin-oliota ottamaan vastaan muuttujat a ja b ja cout tulostaa nuo arvot näytölle. Muuttujia ja muita tässä ohjelmassa olevia piirteitä käsitellään laajasti seuraavissa luvuissa.

Yhteenveto

Tässä luvussa tutkittiin ohjelmaa yksityiskohtaisesti. Opit sisällyttämään tiedostoja käyttäen #include-komentoa sekä käyttämään kommentteja. Opit myös tietämään, mikä on funktio ja kuinka sitä käytetään ohjelmassa.

Kysymyksiä ja vastauksia

K
Mitä #include tekee?
V
Kyseessä on esikäsittelijän ohje ja esikäsittelijää kutsutaan aina ohjelmaa käännettäessä. Juuri tällä ohjeella lisätään ohjeen jälkeen oleva tiedosto koodiin aivan kuin se olisi kirjoitettu sinne itse.

K
Mitä eroa on //- ja /* -tyylisillä kommenteilla?
V
Tuplakauttaviivoin (//) toteutetut kommentit päättyvät rivin lopussa. Kauttaviiva-ja-asteriski (/*) -tyyppiset kommentit ovat voimassa, kunnes kohdataan lopettava kommenttimerkki (*/). Muista, ettei edes funktion loppu päätä /*-tyyppistä kommenttia vaan sinun on sijoitettava kommentin loppuun */-merkit tai muutoin saat aikaan kääntäjän virheen.

K
Mikä erottaa hyvän ja huonon kommentin?
V
Hyvä kommentti kertoo lukijalle, miksi tietty koodi tekee juuri tietyn toiminnon tai selittää jonkun koodilohkon toiminnan. Huono kommentti kertoo, mitä tietty koodirivi tekee. Koodirivit tulisi kirjoittaa siten, että ne puhuvat omasta puolestaan; rivin lukemisen tulisi kertoa rivin tehtävän ilman kommenttejakin.