Sovelto

Johdanto sovelluskehitykseen

Sisältö

- Mitä on sovelluskehitys?
- Modernit ohjelmointikielet
- Millaisia sovelluksia voidaan rakentaa?
- Sovelluksien arkkitehtuurit
- Valmiit järjestelmät, tietokannat ja kirjastot
- Sovelluskehitysprojektit

Sovelluskehitys

(https://fi.wikipedia.org/wiki/Ohjelmointi)

- **Ohjelmointi** tarkoittaa tietokoneelle tai vastaavalle laitteelle jollakin tavalla annettavia toimintaohjeita.
- Tietokoneen suoritin ymmärtää suoraan vain ohjelmia, jotka on kirjoitettu konekielellä. Konekieli koostuu yksinkertaisista operaatioista, esimerkiksi "vähennä rekisterin 3 arvosta rekisterin 5 arvo". Ohjelmointi suoraan konekielellä on mahdollista, mutta hyvin hidasta ja virhealtista. Konekielellä ohjelmia kirjoitetaan vain poikkeustapauksissa.
- Tavallisesti tietokonetta ohjelmoidaan käyttämällä jotakin ohjelmointikieltä. Ohjelmointikielissä käytetään luettavia symboleja ja sovittuja koodeja. Esimerkiksi Pascal-kielisen ohjelman rivi "palkka := palkka-vero;" voisi kääntyä edellä mainituksi konekielen operaatioksi. Ohjelmointikielellä kirjoitettu lähdekoodi joko käännetään konekielelle tai tulkataan kieltä suorittavalla ohjelmalla. Käännettäessä koko ohjelma muutetaan kerralla konekieleksi, minkä jälkeen siihen liitetään tarvittavat kirjastoaliohjelmat. Tämän tuloksena saatu ohjelma voidaan suorittaa ilman kääntäjää. Tulkattaessa käännetään pieni osa kerrallaan (eikä yleensä edes konekielelle asti), suoritetaan se ja käännetään seuraava osa ja niin edelleen. Ohjelman suoritus vaatii siis joka kerta tulkin.

Tietokoneen rakenne

- prosessori
- emolevy
- keskusmuisti (RAM, Random Access Memory)
- massamuisti (levyt, pyörivät, SSD), muistikortit ja –tikut

- näyttö, näppäimistö, hiiri tai muu ohjainlaite (kosketusnäyttö)

Yksiköitä

- bitti, arvo nolla tai ykkönen (bit, b)
- tavu, 8 bittiä (byte, B)
- sana, esim. 32 tai 64 bittiä, riippuu prosessorista
- kilotavu on 1024 tavua (2^{10})
- megatavu = 1024 * 1024 tavua tera = 1024 * 1024 * 1024

Tavun monikerrat					
Binäärijärjestelmä			Kymmenjärjestelmä		
Nimi	Lyhenne	Arvo	Nimi	Lyhenne	Arvo
kibitavu	KiB	2^{10}	Kilotavu	kB	10^3
Mebitavu	MeB	2^{20}	Megatavu	MB	10^6
Gibitavu	GiB	2^{30}	gigatavu	GB	10^9
tebitavu	TiB	2^{40}	Teratavu	TB	10^{12}

Lukujärjestelmät

- kymmenjärjestelmä (desimaali) 123_{10}
- binäärijärjestelmä 1111011_2
- heksa, 16-kantainen $7B_{16}$
- prosessori osaa vain binäärijärjestelmän
- käyttäjät ja ohjelmoijat käyttävät kymmenjärjestelmää (tuttu)
- ohjelmoijat käyttävät heksalukuja kun se helpottaa ohjelmointia, esim 0x20 on 32 kymmenjärjestelmässä ja 100000 binäärisenä
- $00110101 = 0 * 2^7 + 0 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$
  $0 + 0 + 32 + 16 + 0 + 4 + 0 + 1 = 53$
- $2C = 2 * 16^1 + 12 * 16^0 = 2 * 16 + 12 = 44$
  (A = 10, B = 11, C = 12, D = 13, E = 14, F = 15)
- Muistiosoitteissa käytetään poikkeuksetta heksamuotoista esitystä

Tietotyypit

- prosessori käsittelee vain ja ainoastaan binäärilukuja, miten sitten esimerkiksi tekstiä pystytään käsittelemään?
- numeroarvo voidaan tulkita tilanteen mukaan eri tavoin:
  - kokonaisluku
  - liukuluku
  - päivämäärä
  - kirjainmerkki
- Kirjainmerkkien tallennukseen käytetään numeroita ja numero tulkitaan kirjaimeksi käytetyn merkistön mukaisesti
- merkistöjä:
  - ASCII, UTF8, Unicode, IBM850, ANSI, jne…

Merkistöistä

- ASCII kehitetty 60-luvulla, voidaan esittää 127 merkkiä (7-bittiä)
- Unicode (UTF8, UTF16 ja UTF32) yleistynyt viime vuosina ja sillä voidaan esittää 1114112 eri merkkiä
- Unicode:lla pystyy esittämään vaikka kyrilliset aakkoset yhtä hyvin kuin japanilaiset merkit
- Esimerkki: sovellus kirjoittaa tiedostoon tekstiä käyttäen ANSI-merkistöä, sama tiedosto luetaan toiseen sovellukseen joka käyttää ASCII-merkistöä ➔ skandimerkit 'sekaisin'

Käyttöjärjestelmä

- Käyttöjärjestelmä on keskeinen tietokoneen ohjelmisto. Se mahdollistaa muiden ohjelmien toiminnan
  - Windows, Linux, Unix, iOS, Android, MS-DOS, VAX/VMS, Symbian, … n-kpl
- Käyttöjärjestelmän tehtäviä
  - Laitteiston hallinta
  - Tiedostojärjestelmä (File System)
  - Muistinhallinta ja virtuaalimuisti
  - Prosessienhallinta
  - Verkkoprotokollat
  - Käyttäjähallinta ja oikeudet
- Sovelluksen tehdään tietylle käyttöjärjestelmälle, esimerkiksi jos sovellus haluaa luoda uuden tiedoston levyjärjestelmään niin se on pyydettävä käyttöjärjestelmältä
- (.NET ja Java-sovellukset toki toimii kaikilla tuetuilla käyttöjärjestelmillä sellaisenaan, tästä lisää tuonnempana)

Tiedostojärjestelmä ja tiedostot

- Tiedostojärjestelmä määrittää mm. miten tiedostot tallentuvat levylle, kuinka isoja levyjä voidaan käyttää ja miten käyttöoikeudet määritellään
- Hakemistorakenteet ja levyjen nimet/tunnisteet
- hakemistoerottimet
- tiedostonimien muoto ja pituus (voiko käyttää skandeja nimissä?)
- kooditiedostot ovat yleensä käyttöjärjestelmäkohtaisia
- datatiedostot ovat yleensä siirtokelpoisia eri käyttöjärjestelmien välillä niin kauan kun puhutaan jostain yleisestä formaatista, esimerkiksi .txt, .bmp, .jpg, .pdf, jne

Prosessorin toiminta

- prosessori suorittaa konekoodia, esim:
  - hae muistipaikasta X arvo rekisteriin EX
  - lisää arvo 1 rekisterin sisältöön
  - hyppää jos nolla
- yksinkertaistettu malli:
  https://www.youtube.com/watch?feature=player_detailpage&v=cNN_tTXABUA
- Lisää esimerkiksi:
  http://www.mikrobitti.fi/2013/06/nain-toimii-suoritin/
- konekoodi voidaan esittää assemblerina, 'selväkielisenä' konekoodina
- eri prosessoreilla on omat käskykantansa

```
00007FFC2C60816E  mov         rcx,qword ptr [rbp+150h]
00007FFC2C608175  mov         rcx,qword ptr [rcx+1B8h]
00007FFC2C60817C  cmp         dword ptr [rcx],ecx
00007FFC2C60817E  call        00007FFC2C604450
00007FFC2C608183  mov         qword ptr [rbp+78h],rax
00007FFC2C608187  mov         rcx,qword ptr [rbp+78h]
00007FFC2C60818B  cmp         dword ptr [rcx],ecx
00007FFC2C60818D  call        00007FFC853DD618
00007FFC2C608192  mov         qword ptr [rbp+70h],rax
00007FFC2C608196  mov         rcx,qword ptr [rbp+70h]
00007FFC2C60819A  mov         qword ptr [rbp+100h],rcx
00007FFC2C6081A1  nop
00007FFC2C6081A2  jmp         00007FFC2C60824D
00007FFC2C6081A7  mov         rcx,qword ptr [rbp+100h]
00007FFC2C6081AE  mov         r11,7FFC2C4F0088h
00007FFC2C6081B8  cmp         dword ptr [rcx],ecx
00007FFC2C6081BA  call        qword ptr [r11]
00007FFC2C6081BD  mov         qword ptr [rbp+60h],rax
00007FFC2C6081C1  mov         rcx,qword ptr [rbp+60h]
00007FFC2C6081C5  mov         qword ptr [rbp+0F8h],rcx
```

Prosessi

- Prosessi ~ sovellus
- suorituksessa oleva koodi
- moniajokäyttöjärjestelmä suorittaa eri prosesseja vuorotellen lyhyen aikaa kerrallaan
- prosessin sisällä voi olla vielä rinnakkain suorituksessa olevia aliohjelmia, näistä käytetään usein termiä säie (thread)
- Windowsissa prosessin koodi on levyllä .exe –tiedostossa, esimerkiksi Notepad.exe:n käynnistyksessä käyttöjärjestelmä lataa tiedoston keskusmuistiin ja lähtee suorittamaan siinä olevaa koodia

Ohjelmointi

- konekoodia ei juuri (koskaan) enää kirjoiteta
- todella laiteläheinen ohjelmointi tehdään usein assemblerilla
- normaalisti kirjoitetaan korkeantason ohjelmointikielellä ns. source-koodi eli lähdekoodi joka käännetään kääntäjällä konekielelle (binäärikoodiksi) halutulle prosessorille
- jokaisella ohjelmointikielellä on oma kääntäjä
- on olemassa myös tulkkaavia kieliä joissa source-koodia tulkitaan suorituksen aikana eikä varsinaista prosessorikohtaista konekoodia synny lainkaan

Ohjelmointikielten tyypit

- Tyypit
  - proseduraalinen ohjelmointikieli — C/C++, Pascal, Cobol, Fortran, Basic
  - oliopohjainen ohjelmointikieli — C++, Java, C#
  - funktionaalinen ohjelmointikieli — Javascript, Lisp, F#
  - logiikkapohjainen ohjelmointikieli — prolog
- 2000-luvulla olio-ohjelmointikielet (Java, C#) ovat yleistyneet todella paljon ja ovatkin merkittävimmät kielet
- Javascript on paljon käytetty kieli varsinkin selainohjelmoinnissa mutta sitä käytetään myös server-ohjelmoinnissa (Node.js)
- muuten funktionaalisten ja logiikkakielten merkitys on aika pieni
- 80- ja 90-luvuilla tutkittiin paljon tekoälykieliä (logiikka) mutta alkuinnostuksen jälkeen ne ovat jääneet marginaalisiksi kieliksi, nyt tekoäly on taas pop mutta realisoituu hieman eri tavalla kuin edellisen hypetyksen aikana

Kielten piirteitä

- Paradigma
  - proseduraalinen
  - olio-ohjelmointi
  - funktionaalinen
- Syntaksi, kielioppi
  - mitä sanoja ja operaattorimerkkejä voi käyttää ja missä järjestyksessä
  - kääntäjä tarkistaa että syntaksi on kunnossa, virheellinen syntaksi ei kelpaa
- Semantiikka, merkitys
  - miten sovellus toimii ja miten sitä pitää (kannattaa) kirjoittaa/käyttää
- Sekä syntaksiin että semantiikkaan löytyy parhaat käytännöt (Best Practices) ja ne ovat erilaisia eri kielillä

Proseduraaliset kielet

- Basic, Cobol, C/C++, Fortran, Pascal
- koodin rakenne tehdään aliohjelmilla (proseduureilla) joista käytetään myös nimeä funktio
- tietorakenteet (data) määritellään erikseen eikä tietorakenteen määrittelijä voi rajoittaa miten sitä käsitellään

```
/* sovelluksen koodia */
Tili * tili = haeTili("FI710013");
LisaaSaldoa(tili, 10.43);
tili->saldo = -1; /* ??!! */
talletaTili(tili);
}
```

```
/* DATArakenne */
record Tili {
  tilinro : string;
  omistaja : int;
  saldo : money;
}
```

```
/* käsittelyaliohjelma */
void LisaaSaldoa (Tili *tili, money summa) {
  tili->saldo += summa;
}
```

Olio-ohjelmointi

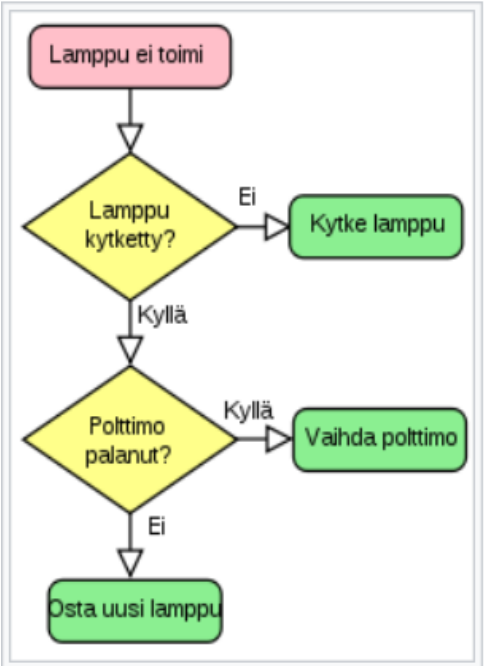
- C++, C#, Java
- koodin rakenne tehdään luokilla ja olioilla. Luokka on määritys ja olio on tilanvaraus koneen muistista sovelluksessa
- data ja sitä käsittelevä koodi on kiinteästi yhdessä ja voidaan määritellä tarkasti käsittelysäännöt

```
/* sovelluksen koodia */
Tili tili = new Tili();
tili.haeTili("FI710013");
tili.LisaaSaldoa(10.43);
tili->saldo = -1; /* ei
onnistu! */
tili.talletaTili();
}
```

```
/* DATArakenne */
class Tili {
  tilinro : string;
  omistaja : int;
  private saldo : money;
void LisaaSaldoa (
 Tili *tili, money summa)  {
  tili->saldo += summa;
}
//…
/* haeTili ja talletaTili –aliohjelmat
…*/
}
```

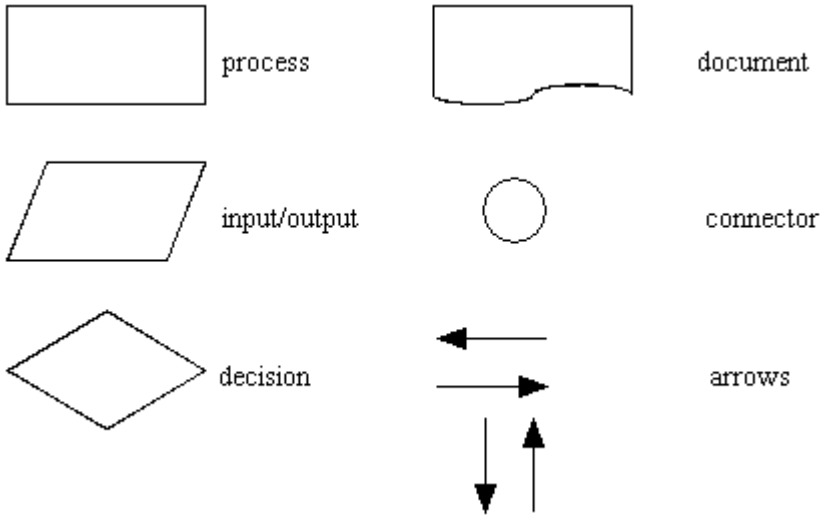
Algoritmi

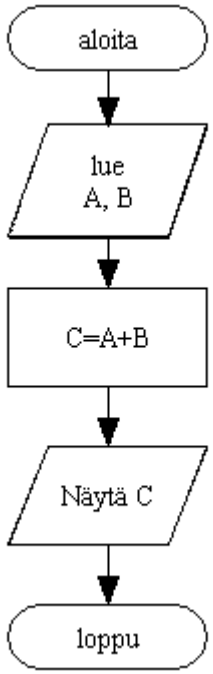
- **Algoritmi** on yksityiskohtainen kuvaus tai ohje siitä, miten tehtävä tai prosessi suoritetaan. jota seuraamalla voidaan ratkaista tietty ongelma.
- tietokone on tyhmä, sille pitää kertoa täsmällisesti mitä pitää tehdä, toiminto määritellään algoritmilla ja kirjoitetaan lopulta jollain ohjelmointikielellä koneen ymmärtämään muotoon
- algoritmi voi olla aivan tekstinä kuvattu toimintosarja tai sitten käytetään esimerkiksi vuokaavioita
- Jos lamppu ei toimi: tarkista että on kytketty
  jos ei niin kytke
  jos on niin tarkista onko polttimo palanut
  jos ei niin hanki uusi lamppu
  jos on niin vaihda uusi polttimo



Vuokaavio

- Vuokaavioita käytettiin varsinkin aikaisemmin paljon kuvaamaan algoritmeja ja tiedon kulkua (kuvat verkkopedakogi.net)





Mihin algoritmeja tarvitaan?

- Sovelluksissa on 1-n kpl erilaisia sääntöjä jotka pitää tarkistaa tai sitten ohjeita miten dataa pitää käsitellä tai muuttaa, nämä ovat sovelluksen algoritmeja
- esimerkiksi henkilötunnuksen tarkistaminen on yksi algoritmi
- henkilötunnuksen muoto: ppkkvvynnnt, jossa
  - ppkkvv on henkilön syntymäaika kaksinumeroisina lukuina päivä, kuukausi ja vuosi; esimerkiksi 010180
  - y syntymä vuosisadan tunnus: 1800-luku: +, 1900-luku: −, 2000-luku: A
  - nnn on yksilönumero, jolla samana päivänä syntyneet yksilöidään. Luku on naisilla parillinen ja miehillä pariton. Yksilönumero on välillä 002–899. Numeroita 900–999 käytetään tilapäisissä henkilötunnuksissa.
  - t on tarkistusmerkki. Se saadaan jakamalla syntymäajan ja yksilönumeron muodostama 9-numeroinen luku (ppkkvvnnn) 31:llä. Jakojäännös, joka on tällöin kokonaisluku väliltä 0–30, muunnetaan tarkistusmerkiksi. Vuosisadan tunnus (y) ei vaikuta tarkistusmerkkiin.
  - tarkistusmerkit: 0123456789ABCDEFHJKLMNPRSTUVWXY
- Harjoitus: tee algoritmi joka tarkistaa että henkilötunnus on oikein

Mitä sovelluksilla tehdään?

- Käytännössä sovelluksilla käsitellään aina jotain dataa (joo, jopa pelit voidaan tulkita näin!)
- Data voi olla
  - tekstiä,
  - taulukollinen numeroita ja muutama diagrammi,
  - pankin asiakkaan tilitiedot,
  - kuva
  - jne…
- Datalla on aina joku formaatti eli muoto ja ko. sovellus osaa juuri sen muodon käsittelyn

Missä data sijaitsee?

- Omalla koneella
  - prosessin muistiavaruudessa
  - esim. tekstitiedostot
  - paikallinen tietokanta (Access, SQL Server, MySQL, Oracle)
- Palvelimella
  - jaetut hakemistot ja sillä olevat tiedostot
  - tietokanta (SQL Server, Oracle, MySQL, …)
- Pilvessä
  - oma hakemisto
  - jaettu hakemisto
  - tietokanta (SQL-kanta tai No-SQL)

Tietorakenteita

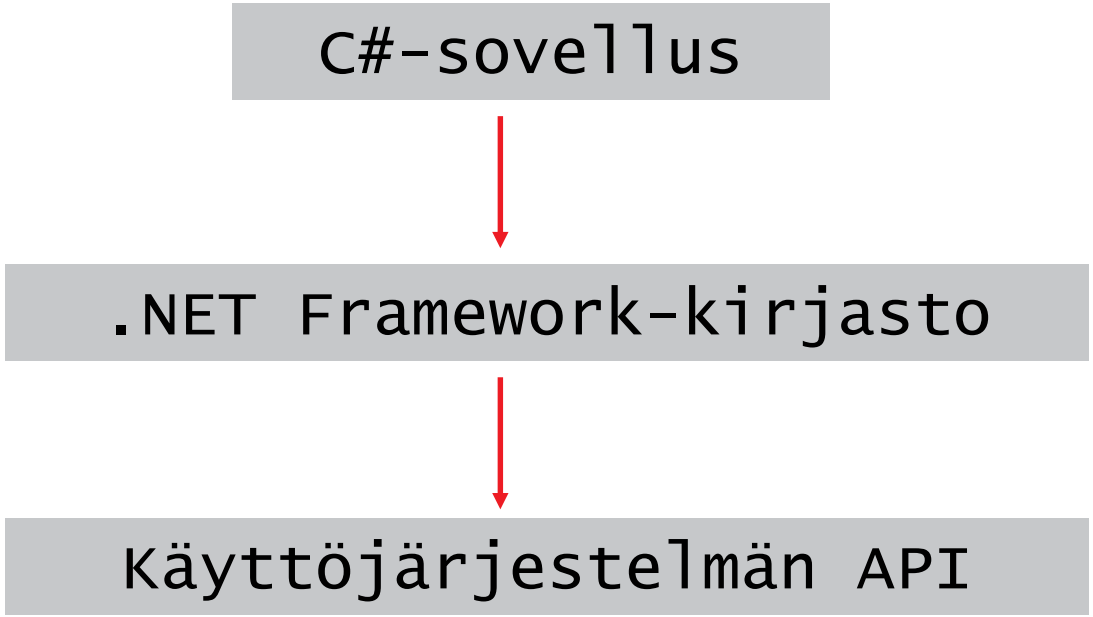
- Sovelluksen suorituksen aikana data on talletettu:
- Skalaarimuuttujiin: kokonaisluku, liukuluku, päivämäärä, merkkijono
- Taulukoihin, n-ulotteiset taulukot, taulukon alkion tietotyyppi määritelty esim. kokonaislukutaulukko
- Rakenteiset muuttujat
  - proseduraalisissa kielissä tietue
  - olio-ohjelmoinnissa luokka (olio)
- Kokoelmaluokat
  - dynaaminen taulukko
  - hashmap/dictionary
  - pino
  - jono

Sovelluksen toiminnasta

- sovellus käynnistyy aina jostain tietystä kohdasta (Entry Point)
- Java, C/C++ ja C#-koodissa se on aina Main-funktio (metodi)
- konsolisovellukset (merkkipohjaiset), ja eräajotyyppiset sovellukset suorittavat Main-funktion ja lopettavat sitten itsensä (prosessi katoaa pois keskusmuistista)
- Käyttöliittymälliset sovellukset, ensin sovellus luo käyttöliittymäikkunan ja jää odottamaan että käyttäjä tekee jotain, esimerkiksi painaa painonappia tai valitsee listasta jonkin rivin, tällöin puhutaan ns. tapahtumaohjatusta rakenteesta
- On paljon erilaisia käyttöliittymäkirjastoja
  - .NET: Windows Forms, Windows Presentation Framework (WPF), Universal Windows Platform (UWP)
  - Java: AWT, Swing

Kirjastot

- Kehitysympäristön (SDK) mukana tulee aina joukko ohjelmankoodikirjastoja jotka tarjoavat sovellusohjelmoijalle käytettävät palvelut
- ohjelmointikielen avulla kirjoitetaan sovelluksen logiikka ja vaikka tietokantakäsittelyn toiminnot löytyvät kirjastoista



Sovellustyypit

- Client (notepad.exe, excel.exe, omasovellus.exe)
- client-server
  - työasemassa toimii sovellus joka keskusteleen palvelimella toimivan sovelluksen kanssa (palvelukerros)
- web-sovellus
  - palvelimella toimii ns. webserver joka palauttaa dataa selaimelle pyynnön mukaan. Tästä puhutaan myöhemmin lisää, mm miten HTTP-protokolla toimii. Selainohjelmointi tehdään javascriptillä.
- mobiilisovellus
  - puhelimessa, Android, UWP, iOS
- sovellusarkkitehtuureja
  - monoliittinen
  - monitasomalli: UI – BL - DA – DB
  - REST, Web Service
  - hybridimallit

Ohjelmointivälineistä

- yleensä käyttöjärjestelmän mukana ei tule mitään ohjelmointivälineitä
- välineet on ladattava (ja sitä ennen ehkä ostettava) ja asennettava
- tarvitset IDE:n ja SDK:n (Integrated Development Environment ja Software Development Kit)
- SDK: kehitysympäristö, kääntäjät ja muut työkalut sekä kirjastot
- IDE: editori, debugger ja liitännät välineisiin (kääntäjä) ja versiohallintaan
- .NET SDK ja Visual Studio tai Code
- JDK ja Eclipse tai NetBeans
- n-kpl muita ympäristöjä…

Muita esille tulevia asioita

- seuraavana siirrytään tutustumaan ohjelmointiin, aloitetaan työasemasovelluksista
- ohjelmointiosuuden jälkeen puhutaan vielä seuraavista asioita:

- Web-sovelluksen toiminta ja ohjelmointi
- Tietokannat, SQL-kannan toiminta
- Testaaminen
- Versiohallinta
- Projektimenetelmät, vesiputousmallista ketteriin menetelmiin
- Pilvipalvelut