

Bevezetés a Programozásba II

1. előadás

Szoftverfejlesztés, programozási paradigmák

© 2014.02.10. Giachetta Roberto
groberto@inf.elte.hu
<http://people.inf.elte.hu/groberto>

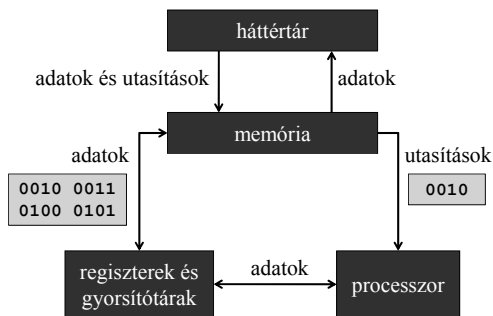
Szoftverfejlesztés

A program

- *A program*
 - *matematikailag*: állapotterek (értékek direktszorzata) felett értelmezett reláció
 - *informatikailag*: utasítások sorozata, amelyek műveleteket hajtanak végre a megadott értékekkel, az *adatokkal*
- A programban foglalt utasítássorozatot, vagy *programkódot* a *processzor* (CPU, GPU, ...) hajtja végre
 - a processzor korlátozott utasításkészlettel rendelkezik, ezért összetett utasításokat nem képes véghezvinni
 - a végrehajtáshoz segédtraktusok (regiszterek, gyorsítótárak) használ, és kommunikál a *memóriával*

Szoftverfejlesztés

A program



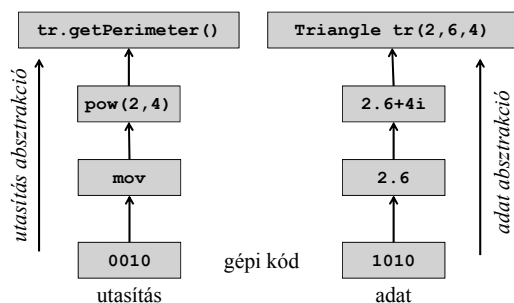
Szoftverfejlesztés

Absztrakció a szoftverfejlesztésben

- A processzor által értelmezhető utasításkészletet és adathalmazt nevezzük *gépi kódnak* (*object code*)
- A programokat ennél magasabb szinten kell elkészíteni, ezért szükségünk van a működés és az adatkezelés absztrakciójára:
 - az *utasításabsztrakció* (*control abstraction*) biztosítja, hogy a processzor egyszerű, egymást követő utasításai (összeadás, mozgatás, összehasonlítás) mellett összetett parancsokat és vezérlési módszert fogalmazzunk meg
 - az *adatabsztrakció* (*data abstraction*) lehetővé teszi, hogy különböző típusokba soroljuk adatainkat, amelyek meghatározzák az értéktartományt, és a véghezhető műveleteket

Szoftverfejlesztés

Absztrakció a szoftverfejlesztésben



Szoftverfejlesztés

Absztrakció a szoftverfejlesztésben

- Az absztrakció előnyei:
 - összetett funkciók könnyebben megfogalmazhatóak
 - el lehet vonatkozni a konkrét megvalósítástól (pl. hardver)
 - a fejlesztési idő jelentősen csökken
- Az absztrakció hátrányai:
 - összetettebb utasításkészletet kell elsajátítani
 - az egyes utasítások végrehajtása nem áttekinthető
 - az absztrakciót jól, körültekintően kell megvalósítani, különben minden rá épülő tevékenység hibás lesz, illetve mellékhatásai lehetnek

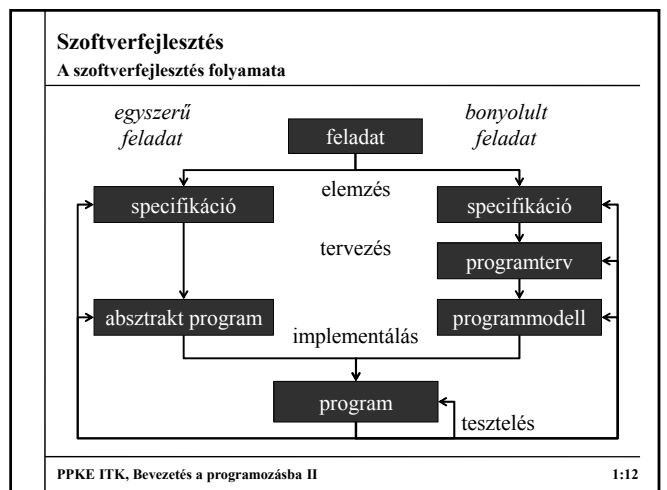
<p>Szoftverfejlesztés A programozási nyelv</p> <ul style="list-style-type: none"> Az absztrakciót megvalósító eszközt nevezzük <i>programozási nyelv</i>nek <ul style="list-style-type: none"> egy adott programozási nyelven megírt programkódot nevezünk a program <i>forráskódjának</i> (<i>source code</i>) a programozási nyelv meghatározza az absztrakció szintjét, a használható típusok és utasítások halmazát, amely egy adott nyelvre rögzített, ám a programozó által általában kiterjeszhető a nyelvet meghatározza a célja, vagyis milyen feladatkörre alkalmazható, továbbá a nyelv rendelkezik egy <i>kifejezőerővel</i>, azaz milyen összetett számításokat képes kifejezni 	1:7
PPKE ITK, Bevezetés a programozásba II	

<p>Szoftverfejlesztés A programozási nyelv</p> <ul style="list-style-type: none"> A programozási nyelvek osztályozása: <ul style="list-style-type: none"> <i>alacsony szintű</i> (assembly): a gépi kódot egyszerűsíti szövegszerűre, de nem biztosít absztrakciót, pl.: <pre> data segment ; adatok number dw -5 ; változó létrehozása data ends code segment ; utasítások ... mov ax, number ; regiszterbe helyezése cmp ax, 0 ; regiszterérték összehasonlítása jge label1 ; ugrás, amennyiben nem negatív mov cx, 0 sub cx, ax ; pozitívra alakítás kivonással ... </pre> 	1:8
PPKE ITK, Bevezetés a programozásba II	

<p>Szoftverfejlesztés A programozási nyelv</p> <ul style="list-style-type: none"> <i>magas szintű</i>: a gépi architektúrától független utasításkészlettel, nagyobb kifejező erővel rendelkezik <ul style="list-style-type: none"> lehetőséget ad az utasítás- és adatabsztrakcióra egy egyszerű reprezentációját adja az alacsony szintű kódoknak pl.: <pre> int main() { int number = -5; // változó létrehozása if (number < 0) // ha negatív number = -number; // ellentettre váltás ... } </pre> 	1:9
PPKE ITK, Bevezetés a programozásba II	

<p>Szoftverfejlesztés Újrafelhasználhatóság</p> <ul style="list-style-type: none"> Az absztrakció során keletkezett összetett funkciókat alacsonyabb szintű funkciók együttes végrehajtását eredményezi <ul style="list-style-type: none"> az így keletkezett összetett funkciót hordozni kell, és későbbi alkalmazásokban újrahasználni Az <i>újrafelhasználható programegységek</i> olyan magas szintű, összetett funkciók, amelyek <ul style="list-style-type: none"> kellően általánosak és hatékonyak alaposan ellenőrzöttek (tesztelve), hibamentesek általában <i>programkönyvtárakba</i> (nyelvi könyvtárba) tömörülnek 	1:10
PPKE ITK, Bevezetés a programozásba II	

<p>Szoftverfejlesztés A szoftverfejlesztés folyamata</p> <ul style="list-style-type: none"> A szoftverfejlesztés a kódoláson túl több lépésből áll, amely függ a feladat bonyolultságától is: <ol style="list-style-type: none"> A feladatot elemezni kell, és megadni a formális megfelelőjét, vagyis a <i>specifikációt</i> A specifikációt alapján megtervezhető a program, amely egyszerű feladatnál az <i>absztrakt program</i>, míg bonyolult feladatnál a <i>programterv</i> elkészítésével jár, amelyből elállítható a <i>programmodell</i> (egyszerűsített célprogram) A tervet implementáljuk a megfelelő programozási nyelven Az implementált programot, illetve a programkódot <i>tesztelésnek</i> vetjük alá, ami módosításokat eredményezhet az implementációban (vagy a korábbi fázisokban) 	1:11
PPKE ITK, Bevezetés a programozásba II	



<p>Szoftverfejlesztés A feladat elemzése és tervezése</p> <ul style="list-style-type: none"> • Önmagában a feladat elemzése is nagyban meghatározza a programfejlesztés folyamatát, ennek két formája: <ul style="list-style-type: none"> • <i>felülről lefelé (top-down)</i>: a főfeladatot részfeladatokra, majd azokat további részfeladatokra bontjuk • <i>alulról felfelé (bottom-up)</i>: a feladatban szereplő egységeket határozzuk meg, majd azokat kombináljuk • A tervezés során egy nyelv-független vázát kell elkészítenünk a szoftvernek, amely megadja annak közeli működését <ul style="list-style-type: none"> • lehet formális, vagy informális modell • használhat modellező eszközt (pl. stuktogram), vagy nyelvet (pl. UML)
<p>PPKE ITK, Bevezetés a programozásba II 1:13</p>

<p>Szoftverfejlesztés Tesztelés</p> <ul style="list-style-type: none"> • A <i>tesztelés</i> annak ellenőrzése, hogy a program teljesíti-e az előírt minőségi mutatókat, célja elsősorban a futási idejű hibák, működési rendellenességek keresése <ul style="list-style-type: none"> • a tesztelés módja szerint lehet: <ul style="list-style-type: none"> • <i>fekete doboz tesztelés</i>: a tesztelendő programrész ismeretlen, csak a hiba voltát fedezzük fel • <i>fehér doboz tesztelés</i>: a programrész teljes mértékben ismert, tehát így a hiba helyét is megtalálhatjuk • <i>szürke doboz tesztelés</i>: a belső adatok, illetve egyes algoritmusok elérhetőek, így a hiba helye behatárolható • a tesztelés módszere lehet <i>statikus</i> (kód kiértékelés és ellenőrzés), illetve <i>dinamikus</i> (futtatás adott tesztesetekkel)
<p>PPKE ITK, Bevezetés a programozásba II 1:14</p>

<p>Szoftverfejlesztés A szoftverfejlesztés optimalizálása</p> <ul style="list-style-type: none"> • A szoftverfejlesztés során a legfőbb cél, hogy a kész program megfeleljen a <i>funkcionális és minőségi követelményeknek</i> <ul style="list-style-type: none"> • emellett, a fejlesztők számára fontos, hogy a kész szoftver fejlesztése a lehető legoptimálisabb legyen • A szoftverek tervezésének és programozásának módszerét nevezzük <i>programozási paradigmának</i> <ul style="list-style-type: none"> • meghatározza a programozási stílust, az absztrakciós szintet • meghatározza az alkalmazható programozási nyelvek körét is, és fordítva • sok programozási nyelv több paradigmát is támogatnak, ezek a <i>multiparadigma</i> nyelvek
<p>PPKE ITK, Bevezetés a programozásba II 1:15</p>

<p>Programozási paradigmák A paradigmák csoportosítása</p> <ul style="list-style-type: none"> • A programozási paradigmákat két csoportba soroljuk: <ul style="list-style-type: none"> • <i>imperatív</i>: a programot állapotváltozások sorozatával írja le <ul style="list-style-type: none"> • az utasításokat szekvenciálisan hajtja végre • megfelel a gépi szintű végrehajtásnak • <i>deklaratív</i>: a program a tartalmát, megjelenését írja le, nem pedig a funkció végrehajtásának módját <ul style="list-style-type: none"> • nem alkalmaz változókat, csak konstans értékeket • az utasításokat nem feltétlenül szekvenciálisan hajtja végre, automatikusan párhuzamosít • magasabb szintű kifejezőerővel rendelkezik
<p>PPKE ITK, Bevezetés a programozásba II 1:16</p>

<p>Programozási paradigmák A jelentősebb paradigmák</p>
<p>PPKE ITK, Bevezetés a programozásba II 1:17</p>

<p>Programozási paradigmák Procedurális programozás</p> <ul style="list-style-type: none"> • <i>Procedurális (Procedural)</i>: <ul style="list-style-type: none"> • a programot <i>alprogramokra (subroutine)</i> bontja, és minden alprogram meghatározott részfeladatot végez el • az alprogramoknak két típusa lehet: <ul style="list-style-type: none"> • <i>eljárás (procedure)</i>: valamilyen utasítássorozatot futtat, végeredmény nélkül • <i>függvény (function)</i>: valamilyen matematikai számítást végez el, és megadja annak eredményét • az alprogramok programkonstrukciókkal épülnek fel, meghívhatnak más alprogramokat, és kommunikálhatnak velük
<p>PPKE ITK, Bevezetés a programozásba II 1:18</p>

Programozási paradigmák	
Procedurális programozás	
<ul style="list-style-type: none"> a vezérlést a főprogram szolgáltatja, az kezeli a teljes programban jelen lévő adatokat nyelvek: <i>Fortran, C, BASIC, Pascal</i> pl. (C++, vektor összegzése függvényel): <pre>int Sum(vector<int> values) { // a függvény paraméterben megkapja a vektort int sum = 0; for (int i = 0; i < value.size(); i++) sum = sum + values[i]; // ciklussal hozzávesszük ez elemeket return sum; // visszatérési érték az összeg }</pre> 	
PPKE ITK, Bevezetés a programozásba II	1:19

Programozási paradigmák	
Strukturált programozás	
<ul style="list-style-type: none"> <i>Strukturált (Structured)</i>: <ul style="list-style-type: none"> a program részegységekre (<i>csomagokra</i>, vagy <i>blokkokra</i>) tagolódik, minden egység rendelkezik egy belépési ponttal, és egy kilépési ponttal a programegységeknek van egy kívülről látható része (<i>interfész</i>), és egy belső megvalósítása (<i>implementáció</i>) a programban használt adatstruktúrák a programegységeknek megfelelően strukturálódnak támogatja a kivételkezelést, tiltja a programkódban történő ugrálást (<i>goto</i>) nyelvek: <i>Pascal, C, ADA</i> 	
PPKE ITK, Bevezetés a programozásba II	1:20

Programozási paradigmák	
Strukturált programozás	
<ul style="list-style-type: none"> pl. (C++, verem típus): <pre>class Stack { private: // rejtett rész, implementáció int* values; // attribútumok int top; public: // látható rész, interfész Stack(){ values = new int[10]; top = 0; } // konstruktor ~Stack() { delete[] values; } // destruktor void Push(int v) { // metódus if (top < 10) { values[top] = v; top++; } } ... };</pre> 	
PPKE ITK, Bevezetés a programozásba II	1:21

Programozási paradigmák	
Objektum-orientált programozás	
<ul style="list-style-type: none"> <i>Objektum-orientált (Object-oriented)</i>: <ul style="list-style-type: none"> a feladat megoldásában az alulról-felfelé megközelítést alkalmazza, alapja az <i>egységbe zárás</i> és az <i>öröklődés</i> a programot egymással kommunikáló objektumok adják, amelyek valamilyen relációban állnak egymással manapság a legnépszerűbb programozási paradigma, a programozási nyelvek jelentős része támogatja objektumorientált támogatással rendelkező nyelvek: <i>C++, Objective-C, Matlab, PHP, Python, Perl, ...</i> tisztán objektumorientált nyelvek: <i>Smalltalk, Java, C#, Eiffel, Ruby, ...</i> 	
PPKE ITK, Bevezetés a programozásba II	1:22

Programozási paradigmák	
Objektum-orientált programozás	
<ul style="list-style-type: none"> pl. (C++, grafikus felületű ablak): <pre>class DemoWindow : public QWidget { // ablak osztály public: DemoWindow(QWidget* parent = 0) { // a konstruktor megkaphatja a szülőt setBaseSize(200, 120); setWindowTitle("Demo Window"); qButton = new QPushButton("Quit", this); // gomb példányosítása } private: QPushButton* qButton; // gomb az ablakon };</pre> 	
PPKE ITK, Bevezetés a programozásba II	1:23