\Chapter{A nyelv definíciója}

\section{Általános szempontok}

A saját nyelv kialakításánál az elsődleges szempont a könnyű kezelhetőség volt, hogy bár több nyelvre, több platformra kerülhet lefordításra a kód, a felhasználó, programozó egyszerűen, könnyen tudja megírni a kódot a definiált nyelven, így nem kell minden célnyelvi szintektikai és szemantikai szabállyal tistába lennie, azokat külön-külön alkalmaznia.

Ezek alapján a nyelv a Python nyelvhez hasonló megoldásokat és jegyeket hordoz magán, a nyelvi szintaktika és szemantika megvalósításának tekintetében.

Ahhoz, hogy a nyelvet definiálni lehessen az alábbiakban áttekintjük a feladat során célnyelvként tekintett nyelveket általánosan, azok megvalósítását, tulajdonságait, fő elemeit. Majd ezután a saját definiált nyelv tervezett nyelvi elemeit vesszük sorra.

\section{A célnyelvek áttekintése}

A C nyelv magasszintű, általános célú programozási nyelv, mely az egyik legelterjedtebb programozási nyelv, a ma fellehető számítógép architektúrák szinte mindegyikére készült C fordító. A nyelv sturkturált és szabványos programozási nyelv, ugyanúgy megtalálhatóak itt a vezérlési szerkezetek mint a többi magas szintű nyelvben. Változó deklarálásakor meg kell adni a változó nevét és típusát is, mivel a nyelv erősen típusok nyelv. A változó neve konvenció alapján betűvel kezdődik és utána betűket és számokat tartalmazhat. Ezután egyenlőségjellel adhatjuk meg a változó értékét. C nyelven az utasításokat a sor végén ;-vel zárjuk le. Mivel az egyes utasítások lezárását ; jelzi, a nyelven a sortörések kihagyhatók, ezek fehér karakterek csak az olvashatóságot növelik, a fordítás során eltávolítására kerülnek. C nyelven írt programok során is az utasítások szekvenciálisan futnak le.

A nyelv nem objektum orientált nyelv, azaz osztályok definíció szintjén nincsenek a nyelvben. Jelen dolgozat további nyelveiben megjelenik ez a funkció, osztályba szervezhető a kód, csakúgy mint a definiált nyelvben is, így valamivel nehezebb C módra történő fordítás. C nyelven a struktúrák találhatók meg, melyek a typedef struct kulcsszóval vezethetnek be. Ebben adhatók meg az osztályokhoz hasonló programelemek. Szintén fontos különbség, hogy a C nyelven megjelennek a pointerek, melyek egy adott változó címere mutató változó. Ezáltal érték szerint és cím szerint is át lehet adni változókat, ilyen más nyelvekben nincs.

C nyelven emellett használható a szelekció, azaz az elágazás. Ezt az if kulcsszó vezeti be, ami után ( és ) között kell megadni az elágazás feltétet. Itt a rendszer azt vizsgálja a megadott feltételek igaz vagy hamis, és aszerint fut le az adott ágban megírt kód. A szelekció többi ágat az else if (feltétel) és az else kulcsszó vezeti be. A C nyelv is ismerni a switch szelekciót, itt is meg kell adni egy kifejezést, változott és a switch belül megadni azon kifejezéseket melyekre az illeszkedést a program vizsgálni fogja.

Emellett a programban megadható iteráció is. Ciklusból a nyelven 3 különböző van, a while, for és do while. Ezek mindegyikéhez kell egy ciklusváltozó, egy feltétel, melyet minden egy ciklusban ellenőriz a rendszer. A for és a while elöl tesztelő ciklus, míg a do while hátul tesztelő, azaz az egyszer biztos le fog futni a cilkusmag, míg az előző kettőben nem biztos. A C nyelvben nincs külön hibakezelés, minden ilyen feladatot a programozónak kell feltételvizsgálattal megoldania, illetve klasszikus esetben nincs automatikus szemétgyűjtő algoritmus sem, a memória felszabadításáról is a programozónak kell gondoskodnia.

A JavaScript egy scriptnyelv, mely leggyakrabban a böngészőben jelenik meg, és kliens oldalon is látható a kódja. A nyelv először MochaScript néven lett létrehozva, majd később a szintaxisa a Java nyelvhez lett hasonló, itt lett a neve JavaScript.

A nyelvet a böngészőkben használt html nyelv statikusságának megtörésére használták, illetve JavaScript segítségével egyszerűen elvégezhető előzetes ellenőrzés az adatokra, mely mellett azonban természetesen mindenképpen kell szerver oldali ellenőrzés is, mivel ahogyan fentebb említettem a JavaScript kód látható böngészőben is különféle eszközökkel, így akár manipulálható is. A kódot a html kódba is ágyazhatjuk, de legtöbbször egy külön .js fáljban van elhelyezve. A JavaScript gyengén típusos nyelv, a változók definiálásakor nem kell típust megjelölni. A változó értéke fogja megadni a változó típusát.

A JavaScriptben is megtalálhatók az ismert vezérlési szerkezetek, ugyanúgy használható az értékadás, szekvencia, szelekció és iteráció is. Sőt még objektum orientálttá is tehető a nyelv. A JavaScriptben külön kifejezéssel lehet változót deklarálni, ez a var kulcsszó, melyet mindig meg kell adni, ezután következik a változó név majd az érték.

Külön meg kell említeni a láthatóságot a JavaScript esetében. Itt a változók alapértelmezetten csak az adott scope-ban elérhetők, ahol létre lettek hozva. Általános esetben mindenképpen egy függvény kerül megírásra, és ebben történik a változók létrehozása, a műveletek elvégzése. A változók csak az adott függvényen belül láthatók ahol létre lettek hozva. Megadhatók azonban olyan módon is a változók, hogy máshonnan is elérhető legyenek, ekkor a függvényen kívül kell deklarálni őket, így globális változókká válnak és elérhetők. Másik módszer ha olyan változóknak adunk értéket amik még nincsenek is deklarálva. A JavaScript ebben az esetben létrehozza a változót, mely azonban globális változóként kerül létrehozásra.

Fentebb említettem a JavaScriptet, mint objektum orientálttá tehető nyelvet. A JavaScriptben általánosságban mindig egy függvényt írunk a function kulcsszóval, még akkor is, ha ez esetleg rejtetten, mondjuk egy jQuery fájlban az alábbi módon nézne ki:

$(„data”).click(function() {

utasítások;

});

Látható a fentebbi kódban is megjelenik a function kulcsszó. A nyelv objetum orientáltsága mindössze annyit jelent, hogy egy osztályként használt függvényt írunk meg, majd ennek adunk adattagokat és további függvényeket is, és végül ezt fogjuk használni objektumként, azaz osztályként.

A JavaScript esetében minden utasítássort ;-vel kell lezárni szintaktikailag, tehát ebben hasonlít az általánosan vizsgált programozási nyelvekhez.

A Pyhton egy objektum orientált programozási nyelv, mely egyre nagyobb teret hódít egyszerű kezelhetősége és rugalmassága miatt. Guido van Rossum a nyelv megalkotásakor a fent emlíett egyszerűséget tartotta a legfontosabbnak.

A nyelv gyengén típusos nyelv, tehát itt sem kell megadni típusmegjelölést, illetve fontos különbség a többi nyelvhez képest, hogy nem használnak ;-t az utasítások lezárásakor, így az utasítások végét a sor vége jelenti. Más magas szintű nyelvekkel ellentétben itt ezért nem is lehetne az egész kódot egy sorban megadni.

Emellett szintén fontos különbség a blokkokat jelző { és } hiánya, a nyelv az olvashatóságot is segítendő, a tabulátorokkal történő behúzással jelzi az egyes blokkokat.

Általánosságban egy adott metódus, osztálydefiníció, elágazás, megírásakor a sor végére :-ot teszünk, ezután a következő sortól tabulátor behízással történik az adott feltétel vagy osztály magjának leírása.

A nyelv támogatja az objektum orientált leírást, a class kulcsszóval bevezetett osztályokat lehet megírni, melyben definiálhatunk változókat, metódusokat majd példányosíthatjuk csak úgy mint az általánosságban vizsgált programozási nyelvek esetében.

A PHP egy szerveroldali nyelv, tulajdonképpen scriptnyelv, az így megírt fájlok futtatásához a fizikai szervergépre valamilyen PHP fordító kell telepíteni és konfigurálni. Legelterjedtebb manapság még mindig az Apache webszerver, mely a UNIX rendszerekre telepíthető.

Érdemes kiemelni a PHP hiba, -illetve figyelmeztetés kijelzését is, mely szerver, az Apache konfigurációs fáljában oldalon, illetve a PHP fájlban is letiltható, illetve engedélyezhető. Érdemes ezt mindig engedélyezni a fejlesztés során, mivel egyéb esetben egyes hibák elfedésre kerülnek a PHP rugalmassága miatt, így a végső eredmény is hibás lehet. Például az alábbi kódban:

<?php

function multiply ($value) {

$value = $value \* 10;

return $value;

}

$retval = multiply();

Print "Return value is $retval\n";

?>

Látható, hogy a multiply függvény egy paramétert vár, és azzal számol, azonban a meghívásakor nem kerül átadásra paraméter. Ilyenkor a PHP rugalmassága miatt, egyszerűen 0 értékkel fog számolni a program, azaz a végeredmény is nulla lesz.

Azonban ez nyilvánvalóan nem a helyes működés lenne (kivéve természetesen, ha pont a 0 számmal szeretnénk meghívni egyébként is a metódust), ezért érdemes bekapcsolni a hibajelzést. Ilyenkor a program kijelzi, hogy bár a metódus deklarálásakor megadtuk, hogy paramétert várunk, meghívásakor nem adtunk neki paramétert.

Általánosságban a PHP megvalósítása miatt, ha ilyen hibákat vétünk (és ha nincs hiba kijelzés) a PHP mindig megpróbálja megoldani, a programot lefuttatni.

A PHP alapvetően nem objektum orientált nyelv. Eredetileg ez szerveroldali scriptnyelv volt, csak a 2004-ben kiadott 5-ös verzióval került be az objektum orientáltság. Mivel a PHP akkoriban a legnépszerűbb szerveroldali scriptnyelv volt, sőt még manapság is az, nagyon sok olyan kód készült, melyben objektum orientált leírás nem volt, így ezeket először át kellene alakítani objektum orientálttá.

A Java nyelven megírt kódok futtatásához is saját környezet kell. A java telepíthető minden operációs rendszerre, valamint komplexitása miatt az egyik legelterjedtebb és leginkább használt nyelv.

A Java nyelvben, ha bármilyen hibát vétünk akkor már a kód fordítása során erről értesítést kapunk, a fordítás pedig leáll. Tehát Java nyelvben a fenti kód (itt természetesen osztályt is kell definiálnunk, majd annak a metódusát meghívni):

public class Test {

private value;

Test(int value) {

this.value = value;

}

public void multiply() {

value = value \* 10;

System.out.println("Return value is: " + value);

}

public static void main(String args[]) {

Test test = new Test();

test.multiply();

}

}

Ez a kód már fordításkor hibát fog jelezni, hiszen a konstruktorban zárójelpárban megadtuk, hogy várunk egy integer értéket, viszont a main függvényben az osztály inicializálásakor nem adtunk meg értéket.

\section{Saját nyelvi elemek}

A saját nyelv definiálásakor a fentiek alapján az alábbi nyelvi szintaktika kerül kialakításra.

A definiált nyelvben is a megadott konstrukciók végrehajtási sorrendjét a vezérlési szerkezetekkel szabályozhatjuk, melyek az értékadásokon kívül a szekvencia, szelekció és iteráció.

A nyelvben az értékadás a Python nyelv egyszerűségét idézi, azaz meg kell adni egy változó nevét és az értékét. Ahogy az utasítások sorának végére, ide sem kell ;-t tenni.

Ha a változó egy adattag egy osztályban akkor a változó definiálásakor meg lehet adni a változó láthatósági módosítóját. Ha nem írjuk ezt ki, akkor alapértelmezetten a változó private módosítót kap, azaz nem lesz kívülről elérhető, csak az osztály metódusai módosíthatják. Emellett megadhatjuk neki a public módosítót, mely esetben mind kívülről, mind belülről az osztályból elérhető és módosítható lesz.

Az értékadást az alábbi példa szemlélteti:

szoveg = „szöveg”

Többszörös értékeadás is lehetséges a nyelvben, azaz egyszerre több vátozó is megaható, melyek aztán felveszik az adott értéket.

ertek = ertek2 = 5

A változók definiálásakor nem kell megadni a változó típusát, a definiált nyelv gyengén típusos, így a változó típusát a változóban tárolt érték fogja meghatározni.

valtozo1 = 5 ~ szám típusú

valtozo2 = "szöveg" ~ text/string tipusú

valtozo3 = valtozo1 + valtozo2 ~ eredmény 5szöveg, mint string típsú változó

A változók, illetve osztály adattagok esetében meg lehet adni láthatósági módosítót is, azonban ez nem kötelező. Ha nincs megadva láthatósági módosító egy adott változóhoz, akkor a nyelvi definicíó alapján alapértelmezetten private láthatóságú lesz a változó, azaz csak az adott osztály, amiben a változó létre lett hozva, az tudja majd kezelni, kívülről nem lehet.

Ha azt szeretnénk, hogy kívülről is lehessen kezelni az adott változót, akkor mindenképpen ki kell írni a láthatósági módosítót, ami public lesz, ha mindenhonnan el lehet érni a változót és protected, ha csak a szülő, illetve a gyerek érheti el. A metódusok belül létrehozott változók minden esetben a metódusokon belül használhatók.

valtozoT = {1, 2, 3, 5, "tizenkettő"} ~ egy adott tömbben többféle típusú elem is szerepelhet, például szám és szöveg. A tömb elemeire a valtozoT(x)-el lehet hivatkozni, ahol az x a tömb x. eleme lesz.

Túlindexelés:

1) Ha a tömb 10 elemű és a tomb(12)-t hívjuk akkor a változóba ahol az elemet letároljuk, vagy metódusba ahova átadjuk egy 0 kerül át

2) Ugyanaz, mint az első esetben, de itt létre is jön a 12. elem. Tehát mondjuk 10 elemű a tömb, a 12.-re elemre való hivatkozással létrejön a 11. és 12. elem, mindkettő 0 értékkel.

Nem tudom melyik a jobb, ha egyáltalán bármelyik is.

tomb = valtozoT + "húsz"

Print(tomb) ~ eredmény: 1 2 3 5 tizenkettő húsz

Del(tomb)

tomb = valtozoT

tomb(2) = 1.25

tomb(3) \*= 5

Print(tomb) ~ eredmény: 1 1.25 15 5 tizenkettő

Del(tomb)

tomb = valtozoT

Print(tomb(7)) ~ eredmény: 0

Del(tomb)

tomb = valtozoT + {5, "nyolc", "miskolc"} ~a már letárolt, névvel ellátott tömbhöz egy névtelen tömböt fűzünk hozzá, ez önmagában nem tárolódik a memóriában, csak az új változóban az eredeti tömbbel együtt, ahhoz hozzáfűzve

valtozoT += {5, "nyolc", "miskolc"} ~ ugyanaz mint az előbb, csak itt nem jön létre új változó, a már meglévőhöz kapcsolódnak az új elemek

Print(tomb) ~ eredmény: 1 2 3 5 tizenkettő 5 nyolc miskolc

Print(valtozoT) ~ eredmény: 1 2 3 5 tizenkettő 5 nyolc miskolc

A második vezérlési szerkezet az elágazás. A definiált nyelvben a szelekció az if kulcsszóval kerül bevezetésre, melyet egy feltétel követ. A feltétel három tagból kell, hogy álljon, mely egy operátor és annak két oldalán valamilyen kifejezés kell, hogy legyen.

Az elágazás további ágait EIf kulcsszóval vezetjük be, mely után szintén egy feltétel kell, hogy álljon, melynek a szerkezete az If feltételével megegyező kell, hogy legyen. Illetve egy utolsó ág van fenntartva arra az esetre, hogy ha a szelekció egyik korába ágának feltétele sem teljesül ezt az Else kulcsszóval kell bevezetni, itt nem kell feltételt megadni.

A szelekció egyes ágain több végrehajtható művelet is helyet kaphat, melyeket a program sorrendben, szekvencia alapján végez el. Az egy adott ághoz tartozó műveleti blokkot a programnyelvben nem kell külön zárójel, illetve kapcsos zárójelpárral körülzárni, a definiált nyelvben az egy adott blokkhoz való tartozást, a programkód tabulálása jelenti, azaz az If kulcsszó majd a feltétel után az utasításokat új sorban, egy tabulátorral beljebb tolva kell megadni, az alábbi példához hasonlóan.

If elso == 1

masodik = elso/2

harmadik = elso-masodik

EIf elso == 2

masodik = elso-1

harmadik = elso+elso

Else

If elso == 42

masodik = elso/2

Else

masodik = elso-20

A szelekció másik típusa a definiált nyelvben olyan elágazás, mely az elágazás elején vár egy változót vagy kifejezést és ezt utána a megadott elemekre próbálja illeszteni. Ezen szelekció a Sw kulcsszóval kerül bevezetésre, melyet követ a változó, illetve kifejezés.

A program futása során az a kódrész fog lefutni, amelyhez tartozó elemre a kifejezés illesztése sikeres volt. Ha nincs ilyen elem akkor egy alapértelmezett kódrészként megadott kód fog lefutni, melyet a Def kulcsszóval kell bevezetni. Ezen kódrész megírása mindenképpen szükséges.

Az egy adott blokkba tartozó kódot ebben a szelekcióban is a tabulátor határozza meg. Az alábbiakban látható erre egy példa.

Sw vizsgált\_elem

eredmeny1:

blokk1\_utasitas1

blokk1\_utasitas2

eredmeny1:

blokk2\_utasitas1

blokk2\_utasitas2

Def:

def\_utasitas

A harmadik vezérlési szerkezet a definiált nyelvben az iteráció. Az iterációt a Loop kulcsszóval kell bevezetni, ezután következik a ciklusfeltétel, majd az utasításblokk, mely az előzőekhez hasonlóan a tabulátor jelzi.

Loop ciklusfeltétel

utasítás1

utasítás2

A ciklus lehet hátul tesztelő is

Loop

utasítás1

utasítás2

ciklusfeltétel

A ciklusfeltétel lehet:

- a < 10 (ha a nem létezik automatikusan létrehozásra kerül 0 értékkel, minden ciklusmag lefutása utána automatikusan növekszik egyel)

- a = 15, a > 10, a-- (15-el jön létre, csökken egyel minden ciklusmag lefutása után és addig fut míg nagyobb mint 10)

Függvények, a funct kulcsszóval kerülnek bevezetésre, amit a függvény neve követ, majd a paraméterek listája.

funct kiir (a)

utasítás

funct add (a, b)

c = a + b

A függvény mindig visszatér az utolsó utasításban megadott változó értékével, ez a függvény meghívásának módja miatt vagy eltárolódik vagy elveszik.

q = add(5,2)

A q-ba a fenti add függvény c értéke kerül, tehát 7.

Lehetséges, hogy a függvény nevét nem kell megadni, a névtelen függvények egyszer fognak lefutni, többet nem lehet őket meghívni, mivel a függvények meghívása a nevükkel történne egyébként.

q = funct (a, b)

c = a + b

c = c / 2

Az osztályok segítségével lehet jól leképezni a valóságban lévő, vagy valósághoz közel álló elemeket is. Nem csak az egyes tulajdonságok és adott elemhez tartozó változók egy egységben történő kezelésére lennének jók, de az egyes elemek, objektumok egymáshoz való viszonyát, leszármazást is meg lehetne ezzel oldani.

Mivel általánosságban is elterjedtek az osztály alapú, objektum orientált nyelvek, ezért a definiált nyelv is ilyen, így a programozóknak, a nyelv használóinak kényelmesebb lesz, mivel olyat nyelvet tudnak használni, melyhez hasonlót már használtak, vagy tanultak róla.

Az osztályok a Create kulcsszóval kell bevezetni, majd az osztály neve következik és a paraméterek listája. Az osztályon belül a paraméterek és metódusok definiálása szintén a tabulátorral van elhatárolva.

Create osztalynev(paramterek)

belső paraméretek definiálása

egyedi metódusok definiálása

Az osztályokban a paraméterekben átadott változókat nem kel külön definiálni, azok automatikusan létrejönnek, és konstruktor is automatikusan lefut. Ezeknek lesz getter és setter metódusa is automatikusan.

Create osztaly(param1, param2)

param3

Oszaly()

param3 = param1 + 15

Ha külön belső változónak szeretnénk konstruktorban értéket adni, akkor külön ki kell írni a konstruktor meghívását, amit az osztály nevével megadott függvénnyel tehetünk meg.

Az osztály változónak módosítása a getter, setter metódusokon keresztül lehetséges például osztaly.getParam1 / osztaly.setParam1 metódus meghívásával. Ugyanígy lehet meghívni saját magunk által írt metódust is.

Egy osztály és azon elemei a memóriában addig maradnak míg arra hivatkozás van, ha az osztály objektuma nincs változóba letárolva, akkor nem marad meg. A ciklusváltozó, ha a ciklusban lett létrehozva akár automatikusan akár manuálisan akkor megszűnik a ciklus végén, a memóriából is törlődik. A programban definiált változók és tartalmuk a program futásának végéig a memóriában maradnak, azután törlődnek. Illetve manuálisan is törölhetők a Del(változónév) segítségével, ekkor törlésre kerül a memóriából az elem.

Ha egy elemre úgy hivatkozunk, hogy nem volt még definiálva, akkor automatikusan létrejön a változó 0 értékkel, ha előtte már volt definiálva és kitöröltük, majd úgy hivatkozunk rá, akkor is 0 értékkel jön létre, bármilyen típusú értéke is volt előzőleg, erre figyelni kell.

% QUEST: A nyelv expression-öket és/vagy statement-eket tartalmaz?

% TODO: EBNF és/vagy szintaxis diagramos nyelvi definíció.