»Mladi za napredek Maribora 2015«

32. srečanje

Interpretacija in pretvorba programskih jezikov

Raziskovalno področje: Računalništvo

Raziskovalna naloga

PROSTOR ZA NALEPKO

Avtor: ŽAN VIDRIH, TOMAŽ LEOPOLD

Mentor: SLAVKO NEKREP

Šola: SREDNJA ELEKTRO-RAČUNALNIŠKA ŠOLA

2015, Maribor

KAZALO

POVZETEK

Program je sproti preverjen s spletnim preizkuševalcem imenovanim RegExr Tester, katerega sta naredila Grant Skinner in skupina »gskinner« z knjižnicami CreateJS in CodeMirror. Ta omogoča pogled kaj naredijo določene fraze, ki jih napišemo. Prav tako omogoča vpogled v živo, torej omogoča sprotne popravke.

Najin program je kombinacija večih RegExpr fraz katere postopoma pretvarjajo kodo.

Primer fraze: /([A-Z])\w+/g. Izbere vse besede v katerih nastopa vsaj1 velika črka.

Program se torej deli v postopke, ki so klicani s pomočjo notranjih funkcij. Če program naleti na kakšno napako se zadnji postopek ne bo izvedel. Dodala sva mu tudi opcijo, da pretvorjeno kodo zaženete neposredno v vašem brskalniku. Ker želiva, da program lahko uporabljalo veliko ljudi z različnimi spletnimi brskalniku sva najino stran oblikovala tako, da deluje brez težav in brez vizualnih napak ne glede na brskalnik.

UVOD

Interpretacija programskih jezikov že od nekdaj predstavlja velik problem. Zaradi velike razlike med samimi sintaksami in logiko je potrebno preverit prav vsak košček programske kode (v nadaljevanju skripte). Ker je na sodobnem trgu takih programov izredno malo, sva se odločila da tudi midva narediva nekaj kar bi lahko pomagalo začetnikom v svetu programiranja kot tudi velikim podjetjem. Osrednji problem pri vsem tem je raznolikost dveh popolnoma različnih programskih jezikov C++ in Javascript. Na začetku sva imela le en cilj - da uspešno pretvori lažje skripte. Predpogoj za uspešno pretvorbo skripte je delujoč program v izvornem jeziku. Pobuda za ustvarjanje takega tipa programa je vsekakor težavnostna stopnja.

- Pretvorbo funkcij iz C++ v Javascript (npr int vsota(int a, int b)->function vsota(a,b)
- Pretvorbo generacije psevdonaključnih števil iz C++ v Javascript (npr. rand()%15 -> (Math.Random()*100000)%15
- Pretvorbo preprostih C++ nizov v Javascript format (npr. int abc[15]; ->abc=new Array(15);
- Pretvorbo C++ funkcij std::cout in std::cin v zelo preprosto Javascript izvedbo (npr int st; cout<< "Vpisi stevilo metov:"; cin>>st; -> alert("Vpisi stevilo metov"); st = uniparse(prompt("st", ""));. Pri tem je vključena uniparse funkcija za lažje procesiranje vhodnih podatkov.

POTREBNO PREDZNANJE

Preden se lahko poglobimo v sam program je potrebno poznati kaj sploh programski jeziki so in na kakšen način delujejo. V nadaljevanju vam bova na kratko predstavila osnove,nastanek in splošen opis programskih jezikov C++ in Javascript,

Ampak če želimo razumeti kaj je sta C++ in Javascript moramo vedeti kaj je to programski jezik

Programski jezik je stroju berljiv umetni jezik, ki je bil razvit, da izraža izračune oz. komputacije, katere lahko izvaja stroj oziroma računalnik. Programski jeziki so nastali zaradi razvoja računalnikov in so bili sprva podrejeni temu razvoju in je zato zgodnje obdobje razvoja programskih jezikov hkrati tudi zgodovina računalnikov.

Vsi programski jeziki imajo nekaj osnovnih gradnikov za opis podatkov in procesov ali pretvorb, ki nanašajo nanje (kot npr. seštevanje dveh števil ali izbira elementa iz zbirke). Ti osnovni gradniki (primitivi) so opredeljeni s sintaktičnimi in semantičnimi pravili, ki opisujejo njihovo strukturo in pomen. Sintaksa je nabor pravil ali formul, ki določa množico (formalno pravilnih) stavkov. Pravila ne omogočajo samo ugotoviti ali je neko zaporedje stavek, določajo tudi samo zgradbo stavka, ki je nujna pri prepoznavanju njegovega pomena. Zato sta sintaksa in semantika tesno povezani.[2]

C++

C++ (**C plus plus**) je splošnonamenski računalniški programski jezik. V C++ so podatkovni tipi statični, zapis kode je prost. C++ omogoča različne programerske pristope in sicer proceduralno, objektno usmerjeno, generično in funkcionalno.

C++ je razvil danski računalnikar Bjarn Stroustrup, v Bellovih laboratorijih.

Razvijati je začel leta 1979 in takrat se je jezik imenoval »*C with Classes*« (C z razredi). Od 90. let je eden najbolj priljubljenih komercialnih programskih jezikov. Najprej so C-ju dodali razrede, nato med drugim virtualne funkcije,preobložitev operatorjev, predloge in rokovanje z izjemami.

Slika 1: Računalnikar Bjarn Stroustrup, http://goo.gl/fzpjyj

C++ se prevede v strojni jezik kar mu omogoča, da enak program izvaja bistveno hitreje od jezikov, ki se interpretirajo ali pa se prevedejo v vmesni jezik (Java, C#, Python, javascript, PHP, ...). Poleg tega da za isto opravilo program porabi manj časa, porabi tudi manj energije, kar je pomembno za naprave z baterijskim napajanjem. C++ se je razvil iz jezika C in programe napisane v jeziku C je razen nekaterih redkih izjem mogoče brez sprememb prevesti s prevajalnikom za C++.

Zgled programa:

```
#include <iostream>


int main()

{

  std::cout << "Pozdravljen svet!" << std::endl;

  return 0;     // ni potrebno po ISO standardu

}
```

Kazalci

Kazalci so najmočnejše orodje v C++. Z njimi se dostopa do pomnilnika, kar omogoča veliko prednosti. Kazalci so v bistvu neke spremenljivke, ki kažejo (navezujejo) na drugo spremenljivko ali objekt. V kazalcu je shranjen naslov objekta ali spremenljivke na katero kaže, s tem se izve njegovo vrednost. Kazalec se deklarira tako, da se najprej napiše vrsto spremenljivke na katero se navezuje, znak za kazalec (»*«), nato ime kazalca in še na koncu znak za dodeljevanje (»=«),predznak (»&«) in ime spremenljivke ali objekta na katero se navezuje(glej spodnji zgled). Težava pri kazalcih pa je, da je to izmed najbolj zakompliciranih snovi v C++. Velika prednost kazalcev je, da ko se prenaša argumente v funkcijo, se prenašajo dejansko sami argumenti ne pa kopije teh argumentov kot pri običajni funkciji.

Znaki v zvezi s kazalci:

- »&« - referenčni kazalec, ki izpiše naslov spremenljivke v RAM-u na katero kaže.
- »*« - dereferenčni kazalec, ki izpiše vrednost spremenljivke na katero kaže.

```
int main(){
 // zgled preproste deklaracije kazalca
 int a;
 int * b = &a // b kaže na a, torej b ima naslov a
 return 0;
}
```

Podatkovni tipi

V programskem jeziku je več podatkovnih tipov, ki so namenjeni različnim stvarem. Vsi tipi imajo svojo velikost v bitih, kar tudi vpliva na njihovo dolžino. Pri deklaraciji teh tipov si pomagamo z določenimi besedami in tako prilagodimo spremenljivko za določeno delo, te besede lahko tudi medseboj mešamo, da dobimo želeno spremenljivko. Zgledi:

- short - je enako, kot da deklariramo navadno spremenljivko (npr. »shor int a« je enako kot »int a«).
- long - poveča velikost spremenljivke, s tem tudi velikost v bitih, kar zavzame več prostora v pomnilniku.
- signed - spremenljivka s tem predznakom lahko predstavlja negativna in pozitivna števila.
- unsigned - spremenljivka s tem predznakom lahko predstavlja samo pozitivna števila, s tem lahko povečamo velikost spremenljivke v pozitivnih številih.

Ime	Opis	Velikost	Doseg
char	Črka ali majhno celo število	1 bajt	signed: -128 to 127
int	integer	4 bajt	signed: -2147483648 to 2147483647
bool	Booleovska vrednost.	1 bajt	true ali false
float	Plavajoča vejica	4 bajt	+/- 3.4e +/- 38 (~7 števk)
double	Dvakrat natančnejša plavajoča vejica	8 bajtov	+/- 1.7e +/- 308 (~15 števk)
wchar_t	Več črk v nizu	2 ali 4 bajt	ena široka beseda

Pogojni stavek

Tudi imenovan: pogojni konstrukt, pogojni izraz ali kontrolni stavek je del programskega jezika, ki na podlagi vhodnih paramterov odloci ali je njihova boolean vrednost resnicna(true) ali napacna(false).

Prav tako lahko to oznacimo kot 1 za true in 0 za false.

Primer:

```
int a = 15;
bool check;

if(a = 15){
  //Ker je pogoj resnicen nastavimo check spremenljivko na true
  check = true;
}else if(a = 13){
  //Pogoj ni resnicen
  check = false;
}else{// ce ni vrednost ne 15 in ne 13 se izvede ta stavek
  check = false;
  std::cout<<"vrednost ni ne 13 in ne 15"<<std::endl;
}
```

Zanke

Zanke so močno orodje v C++ in so večnamenske. Z njimi se rešuje logične in matematične probleme. Poznanih je več vrst zank

- while
- do while
- for

Če je pogoj resničen, se telo funkcije izvaja, dokler pogoj ni več resničen.

Primer while:

```
#include <iostream>
using namespace std;

int main(){
 int a = 0;
 int b = 10;

 while(a<b){
 a++ // a se vsakič ko se zanka izvede poveča za 1
 cout << a << endl; // izpiše vrednost a
 }
 return 0;
 }
```

Javascript

JavaScript je objektni skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani.

Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko sodeluje s HTML-kodo in s tem poživi stran z dinamičnim izvajanjem. JavaScript podpirajo velika programska podjetja in kot odprt jezik ga lahko uporablja vsakdo, ne da bi pri tem potreboval licenco. Podpirajo ga vsi novejši spletni brskalniki.

Opis jezika

Sintaksa jezika JavaScript ohlapno sledi programskemu jeziku C. Prav tako kot C, JavaScript nima vgrajenih vhodno izhodnih funkcij, zato je njihova izvedba odvisna od gostitelja.

JavaScript se veliko uporablja za ustvarjanje dinamičnih spletnih strani. Program se vgradi ali pa vključi v HTML, da opravlja naloge, ki niso mogoče samo s statično stranjo. Na primer odpiranje novih oken, preverjanje pravilnost vnesenih podatkov, enostavni izračuni ipd. Na žalost različni spletni brskalniki izpostavijo različne objekte za uporabo. Za podporo vseh brskalnikov je zato treba napisati več različic funkcij.

Uporaba

JavaScript podpira dve vrsti komentarjev. Enovrstični se začnejo z znakom `//`. Vse, kar je za tem znakom pa do konca vrstice, prevajalnik šteje za komentar. Za komentarje v dveh ali več vrsticah se lahko uporabi drugi način. Znak za začetek komentarja je `/*` in za konec `*/`. Vse, kar je med tema znakoma, je komentar. Prepovedano je gnezdenje komentarjev.

```
// To je enovrstični komentar.

/*
To je daljši komentar,
ki je v dveh vrsticah.
*/
```

Pri definiciji spremenljivke ni treba navesti tudi vrste spremenljivke. Uporabljati jih začnemo tako, da spremenljivki dodelimo neko vrednost ali pa pred imenom spremeljivke uporabimo ključno besedo `var`.

Spodaj je razpredelnica vrst spremenljivk v JavaScriptu. V oklepaju je angleški prevod vrste spremenljivke.

Vrsta	Primer	Opis
niz (String)	"Perkmandeljc"	Niz znakov med dvojnimi ali enojnimi navednicami ("A" je enako 'A').
število (Number)	3.141	Poljubno število brez navednic. Za decimalno ločilo se uporablja pika (ne vejica).
logična spremenljivka(Boolean)	true	Logična spremenljivka ima lahko samo dve vrednosti, in sicer »true« ali »false«.
null (Null)	null	Spremenljivka brez vrednosti.
objekt (Object)		Programski objekt
funkcija (Function)		Definicija funkcije

Sledi nekaj primerov definiranja spremenljivk.

```
var a;
b = -5;
j = "JavaScript";
```

Tabela je sestavljena podatkovna struktura, ki se uporablja za shranjevanje več spremenljivk iste vrste. Do vrednosti spremenljivk dostopamo z indeksom. V JavaScriptu je polje definirano kot objekt, zato ima še dodatne lastnosti.

JavaScript ima operatorje za primerjanje, računanje, prirejanje in logične operacije ter operatorje za posebne namene.

Jezik podpira kontrolne stavke in zanke. Od kontrolnih stavkov se najbolj uporabljajo stavki `if`, `if...else` in `switch` ter zanke `for` in `while`. Oblika zank in delo z njimi je enako kot pri programskem jeziku C.

Funkcija je skupek programske kode, ki se kliče od drugje. Vsaka funkcija ima svoje ime ter lahko sprejema in vrača spremenljivke. Namen funkcij je, da se na enem mestu zbere kodo, ki se večkrat uporablja.

Primer

Pozdravljen svet v JavaScriptu:

```
<script type="text/javascript">
 alert("Pozdravljen svet!");
</script>
```

Program brez HyperText Markup Language ne deluje, torej je potrebno vnest tudi preostali del kode. Ampak kaj sploh je HyperText Markup Language?

Hyper Text Markup Language (slovensko *jezik za označevanje nadbesedila*, kratica **HTML**) je označevalni jezik za izdelavo spletnih strani. Predstavlja osnovo spletnega dokumenta. S HTML razen prikaza dokumenta v brskalniku hkrati določimo tudi strukturo in semantični pomen delov dokumenta.

Enostaven primer:

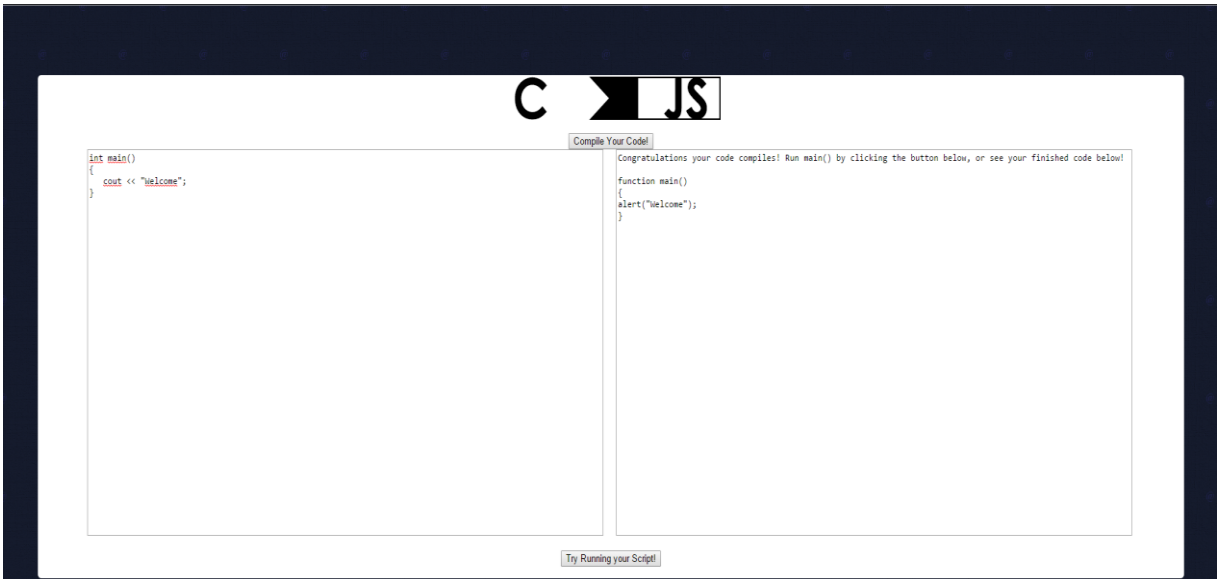
```
<html>
<head>
<title>Primer JavaScripta</title>
</head>
<body>
<p>
            Primer spletne strani.
</p>
<script type="text/javascript">

 alert("Pozdravljen svet!");

</script>
</body>
</html>
```

PODROBEN OPIS RAZISKOVALNE NALOGE

Vsak program je izredno težko obrazložit zaradi raznolikega pomena določene sintakse. Vsak programer si prav tako sam imenuje spremenljivke katere bo uporabljal skozi celoten program. Za opis sva se odločila, da Vam obrazloživa program po vrsticah, ker je program izredno zakompliciran. Trenutno se gostuje na najinem strežniku, tako pa izgleda v brskalniku Google Chrome:



Slika 2: Izgled programa

Program je razdeljen v 2 podokna, v katerem se prikazuje tekst oziroma izpis. V levo okno se prilepi delujočo C++ skripto in ob pritisku na gumb »Compile your Code!« se na desni strani izpiše skripta v jeziku JavaScript.

Ozadje programa

Program je sestavljen iz skripte in HyperText Markup Language. Začetek skripte se označi s <script> in konča s </script>. Preostali del je HyperText Markup Language, katerega lahko s pomočjo Javascripta spreminjamo, dodajamo novi tekst, fotografije ipd.

HyperText Markup Language del

```
<html>

<head>

<link rel="stylesheet" type="text/css" href="i.css">

<meta charset="utf-8" />

<title>Sklanjam</title>

<script src="../jquery.js" ></script>

</head>

<body>

<div class="yolo" align=center>

<div class="logotip"></div>

<button name="TYPE" form="frm" style="margin: 15px auto auto auto;display:block;" onclick="Compile()">Compile Your Code!</button>

<textarea name="TEXTBOX" id="inputbox">

int main(){

   cout << "Welcome";}

</textarea><textarea name="TEXTBOX2" form="frm" id="outputbox" readonly ></textarea>

<br>

<button id="RunMain" onclick="main()" style="margin: 15px auto auto auto;display:block;">Try Running your Script!</button>

</div>

<h5 align="center" style="margin: 15px auto auto auto;">Copyright 2014<br>

Do not distribute links to this web page<br>

Do not copy the contents of this web page<br>

All rights are reserved</h5>

<script id="containerForCode"></script>
```

Javascript del

Skripta programa je zelo obsežna zato ne-bova prikazala celotne naenkrat vendar bova vsako pomembno vrsto posebej razložila.

```
var cpCached = "";   To naredi prazno spremenljivko tipa string
```

-Funkcija piše v tekstovno polje.

```
function append(t,str){
        if(DEBUG)  t.innerHTML = t.innerHTML + str;
}
```

```
var MATHFUNCTIONS = "((sin)|(cos)|(tan)|(atan)|(asin)|(acos)|(atan)|(exp)|(pow)|(sqrt)|(round)|(ceil)|(floor))";
```

-Najde vrstico v kateri se pojavi ena od matematičnih funkcij in jo izbere

```
var PRIMITIVES = "(char|int|short|Byte|Double|float|long|int32|int64|string|signed|bool)";
```

-Preveri vse vrstice in izbere vse navedene tipe spremenljivk

```
var PRIMREGEX = new RegExp("(char(.*?;)|int(.*?;)|short(.*?;)|Byte(.*?;)|Double(.*?;)|float(.*?;)|long(.*?;)|int32(.*?;)|int64(.*?;)|string(.*?;)|signed(.*?;)|bool(.*?;))","ig");
```

- Glavna funkcija ki se izvede ob kliku Compile gumba

```
var FUNCTREGEX = new RegExp(PRIMITIVES+"(?![^{]*})(?![^(]*[)])","ig");
```

- Zagotovi zaščito pred sestujem programa

```
append(tbox,"Step:1(unformatting)\n");
```

- Izpiše »unformatting«

```
r = r.replace(/(^\s.*#|^#).*$/gm,'');
```

- Odstrani simbole

```
r = r.replace(/ +(?=([^"]*"[^"]*")*[^"]*$)/gm, '')
```

- Naredi presledek če pogoj ustreza

Tukaj sledi ogromno izrazov, ki zamenjajo določene znake v C++ skripti.

Zaradi enormne količine jih neboma posebej razlagala.

```
 .replace(/(\r\n|\n|\r|  )/gm,'')
 .replace(/;+(?=([^()]*\([^()]*\))*[^()]*)/gm,'$&\n'
 .replace(/(\{)/gm,'\n{\n');
 r = r.replace(/(}(?!\n))|({(?!\n))/ig,"$&\n");
 try{
 while(r.match(/>>(\w*)>>/igm).length > 0){
 r = r.replace(/>>(\w*)>>/igm,">>$1;\ncin>>");
 alert(r.match(/>>(\w*)>>/igm).length);
 }
 }catch(e){}
 append(tbox,r);
 append(tbox,"\n\n\nStep:2(stripping)\n");
 r = r.replace(PRIMREGEX,"$2$3$4$5$6$7$8$9$10$11$12$13");
 r = r.replace(/^(?=[^+({})-]*?,[^+-]*?$).*?;/gm,"var $&");
 r = r.replace(FUNCTREGEX,"function ");
 r = r.replace(new RegExp(PRIMITIVES,"ig"),"");
 r = r.replace(/(usingnamespacestd;)|/gi,"");
 r=r.replace(/(return|if|else|catch|goto|for|while|do|try)+(?=([^"]*"[^"]*")*[^"]*$)/igm,"$
 & ");
```

```
r = r.replace(/(^.*srand.*$)/gm,"");
r = r.replace(/rand\(\)/gi,"(Math.random()*100000)");
append(tbox,r);
append(tbox,"\n\n\nStep:3(refactoring)\n");
r = r.replace(MATHREGEX,"Math.$&");
r = r.replace(/<<([^<>;"]*?[+\/*%-][^<>])*?(?=<|;)/ig,"+($1)");
r = r.replace(/<</g,"+")
.replace(/\+endl/gi,"")
.replace(/\[(\d.*?)]/g,"=new Array($1);")
.replace(/cout;\n/g,"");
r = r.replace(/cout\+/g, coutOverride);
r = r.replace(/;[\s;]*(;)/ig,"$1");
r = r.replace(/;\d*?(;)/,";");
r = r.replace(/^,|^;/igm,"");
r = r.replace(/(function \w*\(.*\))([^{]*?[\w\d])/igm,"$1{$2")
append(tbox,r);
var cinTriggered = false;
append(tbox,"\n\n\nStep:4(modifying(javascript flavour tweaks))\n");
var lines = r.split("\n");
```

```
for(var i = 0;i<lines.length;i++){
  if(lines[i].indexOf("cin>>") != -1){
    cinTriggered = true;
    lines[i] = lines[i].replace(/cin>>/,"");
    lines[i] = lines[i].substring(0,lines[i].length-1);//strip semicolon
    lines[i] = lines[i] + " = uniparse(prompt(\""+lines[i]+"\", \"\"));";
    continue;}
  if(lines[i].split("(").length - 1 != lines[i].split(")").length -1)
    lines[i] = lines[i].substring(0,lines[i].length-1)+");";
  if(lines[i].split("[").length - 1 != lines[i].split("]").length -1)
    lines[i] = lines[i].replace(/\]/,"");}
r = lines.join("\n");
if(cinTriggered == true)r = "function uniparse(str)";
 CODE = r;
 append(tbox,r);
 append(tbox,"\n\n\nStep:5(Check for mistakes)\n");
 try {
        eval(CODE);
tbox.innerHTML = ("Congratulations your code compiles! Run main() by clicking the button below, or see your finished code below!\n\n"+r);
document.getElementById("containerForCode").innerHTML = CODE;
document.getElementById("RunMain").onclick = main;
} catch (e) {
if (e instanceof SyntaxError) {
  append(tbox,e.message);
 }
 }
```

METODOLOGIJA DELA

Program sva izdelovala postopoma glede na težavnostno stopnjo. Začela sva pri lahkih skriptah in kasneje prešla na zahtevne skripte. Ker ima vsak programer svoj način dela mora biti program najbolj optimiziran da prenese takšno težavnost.

REZULTATI

Program sva sproti testirala s pomočjo najinih sošolcev. Uporabljali so enostavne programe, katere so naredili sami. Vse odkrite težave sva sproti rešila. C-družinski jeziki so si podobni in se da med njimi z lahkoto prevajat, dijakom prehod iz C++ v Javascript ni predstavljal težav.

INTERPRETACIJA REZULTATOV

Ker je raziskovalna naloga skoraj inovacija in ni podobnih programov na sodobnem trgu je skoraj nemogoče primerjati dva izdelka. Delovne hipoteze so bile potrjene že med samim ustvarjanjem seminarske naloge. Sam program bi lahko in še bova izboljšala s dodajanjem podpore za nove funkcije in zapletenejše zanke.

ZAKLJUČEK

Program ima veliko možnosti za nadaljnji razvoj na vseh področjih. Se vedno dopolnjuje in popravlja. Ker programski jeziki niso vezani na državo je lahko to internacionalni projekt, kjer bi lahko sodelovale tudi druge države. Skripte je izredno malo za tako obširen program. Problematika programa je lahko izpopolnjevanje v nedogled, saj se programski jeziki spreminjajo in preoblikujejo. Prav tako se dodajajo nove stvari, ki so poznane le najbolj strastnim uporabnikom. Prvoten namen projekta je bil že zdavnaj dosežen, sedaj se le postavlja na višji nivo.

VIRI

- Žumer, Viljem; Brest, Janez (2004). *Uvod v programiranje in programski jezik C++*. Maribor: FERI. Pridobljeno dne 10-2-2015.

- Juvan, Martin; Zaveršnik, Matjaž (2000). *Vaje iz programiranja: C, C++ in Mathematica*. Ljubljana: Študentska založba. Pridobljeno dne 10-2-2015.

- Milan Podbrščček, Janko Harej, Andreja Vehovec, Andrej Florjančič, Tea Lončarič, Saša Divjak. "Jeziki za vizualno programiranje". Projekt e-računalništvo, Ministrstvo za šolstvo. Pridobljeno dne 10-2-2015.

- http://www.regexr.com/ Pridobljeno dne 10-2-2015.