|  |
| --- |
| Universitatea „Ștefan cel Mare” Suceava |
| Toy – limbaj de programare |
| Proiect laborator „Proiectarea compilatoarelor I” |

|  |
| --- |
| Alexandru Dascălu  1/21/2016 |

Cuprins

[Arhitectură proiect 2](#_Toc454041461)

[Structură director 2](#_Toc454041462)

[Exemple corecte 2](#_Toc454041463)

[Exemple greșite 2](#_Toc454041464)

[Creare 3](#_Toc454041465)

[Interpretor 3](#_Toc454041466)

[Structură limbaj 4](#_Toc454041467)

[Tipuri de date: 4](#_Toc454041468)

[Afișare ecran: 5](#_Toc454041469)

[Instrucțiuni: 5](#_Toc454041470)

[Operatori matematici 6](#_Toc454041471)

[Operatori relaționali 6](#_Toc454041472)

[Funcții aritmetice 6](#_Toc454041473)

[Funcții trigonometrice 6](#_Toc454041474)

[Funcții proprii (conversii distante): 7](#_Toc454041475)

[Teste 7](#_Toc454041476)

[Utilizarea funcții trigonometrice și calcularea ariei unui cerc. 8](#_Toc454041477)

[Utilizarea structurii for și funcții matematice 8](#_Toc454041478)

[Bibliografie și resurse 9](#_Toc454041479)

# Arhitectură proiect

„Toy” este un compilator/interpretor al limbajului de programare cu același nume. După cum sugerează și numele, „toy” este un limbaj de programare de test. Aplicația care interpretează codul a fost generată cu ajutorul aplicațiilor windows **win\_flex** și **win\_bison**.

Codul scris este interpetat de aplicație și este executată o secvență de cod C corespunzătoare. De asemenea, se obține și codul echivalent în C plus executabilele.

## Structură director

* 5\_ex\_corecte: conține 5 exemple de cod corect
* 5\_ex\_gresite: conține 5 exemple de cod greșit
* src: contine codul sursă
* documentatie – documentul curent
* readme.md – indicații scurte
* ruleaza\_teste\_ok.bat – fișier batch – execută testele corecte și crează fișierele aferente de cod C și executabilele
* ruleaza\_teste\_not\_ok.bat – fișier batch – execută testele gresite

### Exemple corecte

* test1\_ok – operații cu numere reale: logaritm, rotunjire prin adaus, rotunjire prin lipsa, operații aritmetice elementare, testare stop
* test2\_ok – tipuri de date, sumă, înmulțiri, diferențe, împărțiri,
* test3\_ok – funcții proprii (conversii între unități de masură), modul, putere
* test4\_ok – operatii trigonometrice (sinus, cosinus), testare constante, aria unui cerc, ridicare la putere
* test5\_ok – factorial, structuri de control: instrucțiunea FOR.

### Exemple greșite

* test1\_not\_ok: greșit pentru că folosește „int” in loc de „intreg” pentru tipul de dată număr întreg
* test2\_not\_ok: greșit pentru că folosește cuvinte din afara lexicului „exemplu este gresit”
* test3\_not\_ok: greșit pentru că folosește altă instrucțiune pentru stop
* test4\_not\_ok: greșit pentru că se adună un cuvânt cu un întreg
* test5\_not\_ok: sintaxa FOR greșită

## Creare

toy.exe

GCC

## Interpretor

Modalități de test:

1. In linia de comandă: **toy.exe**. Se deschide interpretorul toy și fiecare linie scrisă va fi executată pe loc. Toate instrucțiunile corecte vor fi translatate în C în fișierul cod.c.
2. In linia de comanda: **toy.exe nume \_sursă**. Se va compila și executa fișierul sursă linie cu linie, de la dreapta la stânga. Codul C va fi scris în fișierul cod.c
3. In linia de comanda: **toy.exe nume\_sursă nume\_output.** Se va compila și executa fișierul sursă linie cu linie, de la dreapta la stânga. Codul C va fi scris în fișierul nume\_output.c.

Procesul este relativ simplu: executabilul toy.exe generat de compilatorul GCC, poate să primească input de text de la tastatură sau dintr-un fișier text. Textul va fi împărțit în lexeme și în funcție de fiecare simbol se va scrie în fișierul de output codul echivalent în C și se va executa o anumită secvență de cod asociată expresiei respective.

# Structură limbaj

Limbajul „toy” (după cum sugerează și numele) este un limbaj de programare formal. Nu este foarte solid și are un număr limitat de funcționalități.

Toy 🡪 C

|  |  |
| --- | --- |
| Toy | C |
| stop | return 0: |

## Tipuri de date:

|  |  |
| --- | --- |
| Toy | C |
| var, întreg, real | * Double, float, int |

## Afișare ecran:

|  |  |
| --- | --- |
| Toy | C |
| ->( „text”) | printf(”%s” „text”) |

## Instrucțiuni:

|  |  |  |
| --- | --- | --- |
| Instrucțiune | Toy | C |
| Instrucțiunea de atribuire | nume\_variabila=valoare  EX: x=2 | Nume\_var = valoare;  EX: a=1; |
| Instructiunea IF | **?** expresie instructiune 1 **??** instructiune 2  EX: ? x< 2 ->”Da” | If expresie\_booleana:  Instructiune1  EX: if x < 2:  Print „Da” |
| Instrucțiunea IF - ELSE | **?** expresie instructiune 1 **??** instructiune 2  EX: ? x< 2 ->”Da” ?? -> „Nu” | If expresie\_booleana:  Instructiune1  else:  Instructiune2  EX: if x < 2:  Print „Da”  else:  Print „Nu” |
| Instrucțiunea FOR | **de (**expr **ori):** instructiune  EX: var i  **de (5 ori): -> i**  **executa** | **For (int i = 1; I < 5; i++)**  instructiune  EX**:**  **For (int i = 1; I < 5; i++)**  **Printf(„%d ”, i);** |
| Instrucțiunea WHILE | **cat\_timp** conditie **executa**  instructiuni | **while**(conditie):  instructiuni |

## Operatori matematici

|  |  |
| --- | --- |
| Toy | C |
| + | + |
| - | - |
| \* | \* |
| / | / |
| %, mod | % |

## Operatori relaționali

|  |  |
| --- | --- |
| Toy | C |
| > | > |
| < | < |
| == | == |

## Funcții aritmetice

|  |  |
| --- | --- |
| Toy | C |
| Putere(a, n) | pow (a,n) |
| rad a | sqrt (a) |
| Log2(a) | loge (a, 2) |
| Log10(a) | log10(a, 10) |
| |sus (a) –rotunjire adaugare | ceil(a) |
| |jos (a) – rotunjire scadere | floor(a) |
| !(a) – factorial | factorial(a) |
| modul (a) | abs (a) |

## Funcții trigonometrice

|  |  |
| --- | --- |
| Toy | C |
| Sin (a) | sin(a) |
| Cos (a) | cos(a) |
| Tan(a) | tan(a) |
| Sinh(a) | sinh(a) |
| Cosh(a) | cosh(a) |
| Tanh(a) | tanh(a) |

## Funcții proprii (conversii distante):

|  |  |
| --- | --- |
| Toy (fișierul func.h) | C (fișierul func.h) |
| m\_>>\_km(a) | func.m\_la\_km(a) |
| km\_>>\_m(a) | func.km\_la\_m(a) |

# Teste

Directoarele 5\_ex\_corecte și 5\_ex\_greșite conțin codul sursă pentru test. Se pot executa automat folosind batch-urile corespunzătoare:

* **ruleaza\_teste\_ok.bat**
* **ruleaza\_teste\_not\_ok.bat**

## Utilizarea funcții trigonometrice și calcularea ariei unui cerc.

|  |  |
| --- | --- |
| Toy | C |
| intreg nrNeg = -1  ->("Numar\_negativ")  ->(nrNeg)  ->("Modul")  var nrPositiv = modul(nrNeg)  ->(nrPositiv)  de (nrPositiv ori): nrPositiv = nrPositiv + 1  ->("Rezultat\_for")  ->(nrPositiv)  var fac = fact(3)  ->("Factorial")  ->(fac)  ->("Factorial2")  ->(fact(fac))  stop | #include <stdio.h>  #include <math.h>  #include "../src/func.h"  int main ()  {  int nrNeg = - 1.00;  AFISEAZA ( "Numar\_negativ" ) ;  AFISEAZA ( nrNeg ) ;  AFISEAZA ( "Modul" ) ;  double nrPositiv = abs ( nrNeg ) ;  AFISEAZA ( nrPositiv ) ;  int i; for (i = 1; i < i < 5; i++)  nrPositiv = nrPositiv + 1.00;  AFISEAZA ( "Rezultat\_for" ) ;  AFISEAZA ( nrPositiv ) ;  double fac = factorial ( 3.00 ) ;  AFISEAZA ( "Factorial" ) ;  AFISEAZA ( facc ) ;  AFISEAZA ( "Factorial2" ) ;  AFISEAZA ( factorial ( fac ) ) ;  getchar();  return 0;  } |
|  |  |

## Utilizarea structurii for și funcții matematice

|  |  |
| --- | --- |
| Toy | C |
| intreg nrNeg = -1  ->("Numar\_negativ")  ->(nrNeg)  ->("Modul")  var nrPositiv = modul(nrNeg)  ->(nrPositiv)  de (nrPositiv ori): nrPositiv = nrPositiv + 1  ->("Rezultat\_for")  ->(nrPositiv)  var fac = fact(3)  ->("Factorial")  ->(fac)  ->("Factorial2")  ->(fact(fac))  stop | #include <stdio.h>  #include <math.h>  #include "../src/func.h"  int main ()  {  int nrNeg = - 1.00;  AFISEAZA ( "Numar\_negativ" ) ;  AFISEAZA ( nrNeg ) ;  AFISEAZA ( "Modul" ) ;  double nrPositiv = abs ( nrNeg ) ;  AFISEAZA ( nrPositiv ) ;  for (int i = 1; i < i nrPositiv; i++)  nrPositiv = nrPositiv + 1.00;  AFISEAZA ( "Rezultat\_for" ) ;  AFISEAZA ( nrPositiv ) ;  double fac = factorial ( 3.00 ) ;  AFISEAZA ( "Factorial" ) ;  AFISEAZA ( fac ) ;  AFISEAZA ( "Factorial2" ) ;  AFISEAZA ( factorial ( fac ) ) ;  getchar();  return 0;  } |

# Bibliografie și resurse

* <https://www.gnu.org/software/bison/>
* <http://gnuu.org/2009/09/18/writing-your-own-toy-compiler/>
* <http://stackoverflow.com/questions/19200808/parsing-a-while-loop-in-bison>
* <https://github.com/Package/Bison-Flex-Calculator>
* <http://stackoverflow.com/questions/25709334/output-ast-from-flexbison-to-main-cpp>