Upute za 1. laboratorijsku vježbu

1. Uvod

Ovo je prva u nizu uputa za obavljanje laboratorijskih vježbi iz kolegija Jezici i prevoditelji, koji se predaje na FESB-u, smjer 220 i 250. Postoji 6 vježbi, koje će vas naučiti jeziku Cool (eng. *Class room object oriented language*), te korištenju programa Flex i Bison. Nakon vježbi student ako želi može nastaviti sa izradom seminarskog rada, koji predstavlja izradu prevoditelja za jezik Cool.

Upute su pisane u obliku tutoriala, a svi potrebni materijali se nalaze na elearning portalu. Vježbe se rade u već pripremljenoj virtualnoj mašini, koju je potrebno skinuti i instalirati na računalo. Na kraju vježbi studenti će trebati predati pismeni izvještaj o vježbama.

Ukoliko u uputama naiđete na kakvu grešku, molim vas da na email **sikora@fesb.hr** pošaljete poruku sa opisom o čemu je riječ, kako bih to mogao ispraviti. Na taj način ćete pomoći budućim generacijama studenata.

1.1 Postavljanje okruženja za izradu seminarskog rada

Za laboratorijske vježbe i izradu seminarskog rada, trebate najprije postaviti okruženje u kojem možete izraditi kompajler. Zbog složenosti instaliranja svih potrebnih alata, pripremljena je unaprijed konfigurirana Linux virtualna mašina (VM), koja se koristi putem programa **Oracle VirtualBox**. Napomena: da bi mogli koristiti virtualnu mašinu trebate u BIOS-u uključiti mogućnost virtualizacije na svom procesoru.

1.2 Postavljanje virtualne mašine

Ideja VM-a je da pokrenete zasebno virtualno računalo sa svojim operativnim sustavom unutar programa za virtualizaciju, u ovom slučaju **VirtualBox**-a. Ovakvo virtualno računalo i operativni sustav koriste vaše stvarno računalo, ali ne izravno, nego posredstvom programa za virtualizaciju.

Za ovaj kolegij pripremljena je slika VM-a, sa obavljenim svim podešavanjima i instaliranim svim programima za izradu kompajlera za **Cool**. Kada je instalirate, VM će uzeti 1GB RAM-a, plus 10 GB prostora na tvrdom disku.

1.3 Instalacija VirtualBoxa

Najprije je potrebno preuzeti i instalirati program za virtualizaciju - **Oracle VirtualBox**. Program skinite sa: https://www.virtualbox.org/wiki/Downloads i instalirajte ga.

1.4 Instalacija VM slike za izradu seminarskog rada

Preuzmite sliku VM-a sa elearning sustava. Veličina je približno 8 GB. Raspakirajte datoteku u direktorij po svom izboru. Nakon što ste raspakirali VM, dvaput kliknite na datoteku ***JIP - Ubuntu 22 x64.vbox*** (plava ikona - *Virtual Box Machine Definition*) i VM će se otvoriti u **VirtualBox**-u.

1.5 Dodatne mogućosti

Sve što je nužno za rad u vježbama ste napravili. Ovdje vam donosimo još jednu stvar koja nije nužna ali vam može pomoći u radu. Naime, zgodno je stvoriti dijeljeni folder virtualne mašine i vašeg računala.

- ugasite virtualnu mašinu
- na računalu napravite folder
- u Oracle VM VirtualBox desnom tipkom kliknite na **JIP-Ubuntu 22x64** virtualnu mašinu i izaberite **Settings/Shared Folders**
- kliknite na ikonicu sa znakom plusa
- u okviru **Add Share** pod **Folder Path** izaberite novo stvorenu mapu i pod **Folder Name** stavite **`shared_on_host`**
- pokrenite virtualnu mašinu i otvorite terminal
- mountajte disk sa dijeljenom mapom pomoću naredbe: **`sudo mount -t vboxsf shared_on_host shared_on_VM`** (adminstratorska lozinka je **`coolfesb`**)
- pomoću naredbe **`cd shared_on_VM`** uđite u dijeljenu mapu

Sada možete koristiti ovu mapu za prebacivanje datoteka između vašeg računala i virtualne mašine.

Studenti koji rade na ARM arhitekturi ne mogu koristiti virtualnu mašinu, ali imaju na raspolaganju online COOL interpreter https://nathanfriend.io/cooltojs/

1.6 Cool primjeri

Kada instalirate VM, možete početi raditi sa jezikom Cool. Korisnički račun je **`student`**, a lozinka je **`coolfesb`** . Sve datoteke vezane uz ovaj seminarski zadatak nalaze se u folderu **`jip`**

Primjeri Cool programa nalaze se u direktoriju **`examples`**. Da biste kompajlirali primjer, pozovite u terminalu naredbu **`coolc`** sa odgovarajućim parametrima. ukoliko je sve u redu, kompajler će stvoriti izvršnu datoteku s nastavkom **`.s`**, koju možete pokrenuti pomoću simulatora **MIPS** procesora **`spim`**.

Na primjer, da biste kompajlirali i pokrenuli ***hello_world.cl***, u terminalu, u folderu **`jip/examples`** upišite slijedeće naredbe:

```
coolc hello_world.cl
spim -file hello_world.s
```

Na ekranu će se ispisati:

```
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
The following symbols are undefined:
main
```

```
Hello, World.
Cool program successfully executed
```

Prvi dio ispisa je zaglavlje koje ispisuje sami simulator **spim**. Neka vas ne zabrinjava poruka da nema simbola **`main`** – poruka je posljedica greške u Spim simulatoru, a ne vaše greške. Nakon toga je ispis programa, a na kraju **spim** simulator ispisuje je li program uspješno izvršen.

1.8 Programski jezik Cool

Cool je kratica od *Classroom Object Oriented Language*. Ovaj programski jezik je dizajniran tako da se kompajler za Cool može napraviti u jednom semestru. Neka rješenja u jeziku Cool čine ga nepraktičnim za korištenje u stvarnom svijetu, a takva rješenja su odabrana baš zato da bi se lakše pisao kompajler za Cool. Zanimljiva je činjenica da je broj Cool kompajlera puno veći od broja Cool programa.

Najvažnije komponente jezika Cool su apstraktni tipovi podataka, statičko određivanje tipova, nasljeđivanje i upravljanje memorijom (eng. *garbage collector*). Iz jezika su izostavljene neke stvari radi lakše izrade kompajlera. Kao dodatnu dokumentaciju za jezik možete koristiti priručnik **cool_manual.pdf** sa elearning portala.

Za editiranje koda u VM-u imate na raspolaganju editor **gedit**, ali vam preporučamo rad u okolini **Visual Studio Code (VSC)**. U njoj imate instaliran i dodatak za *syntax higlitning* pa će vam rad biti ugodniji. Nju je najlakše pokrenuti sa **code .** pa će nakon toga VSC biti pokrenut u vašem trenutnome folderu. Sa **ctrl+J** možete otvoriti terminal unutar VSC-a i nastaviti sa radom.

1.9 Klasa Main

Programi u Cool jeziku su tekstualne datoteke sa nastavkom ***.cl***. Sa nekim od editora napravite novu datoteku **`test.cl`** u folderu **`/cool/examples`**.

Svaki program mora imati klasu **`Main`**:

```
class Main {
      ...
};
```

Deklaracija klase počinje ključnom riječi **`class`,** sadržaj klase je obuhvaćen vitičastim zagradama, a klasa završava znakom **;**

Cool program je lista klasa.

1.10 Metoda main()

Klase mogu sadržavati metode. Svaka metoda mora imati ime, popis argumenata u zagradama i listu izraza koji čine tijelo metode unutar vitičastih zagrada. Svaka metoda treba biti završena sa znakom **;**

Svaki program mora obavezno imati metodu **`main()`** klase **`Main`** :

```
class Main {
    main() {...};
};
```

Metodom **`main()`** započinje izvršenje programa. Ova metoda ne smije imati argumente.

Sadržaj metode je lista izraza, navedena unutar vitičastih zagrada. U jeziku Cool nema ključne riječi **`return`** za određivanje koju vrijednost vraća metoda. Vrijednost metode je vrijednost zadnjeg izraza u tijelu metode.

Za početak ćemo staviti u metodu **`main()`** samo jedan izraz:

```
class Main {
    main() { 1 };
};
```

1.11 Prevođenje i pokretanje

Ovaj program ćemo prevesti sa **coolc**:

```
"test.cl", line 2: syntax error at or near '{'
Compilation halted due to lex and parse errors
```

Prevođenje nije uspjelo. Vidimo da postoji sintaksna greška. Razlog je šta metode moraju imati tip. Odrediti ćemo tip metode **`Main`** kao **`Int`.**

```
class Main {
    main():Int { 1 };
};
```

Ponovo ćemo prevesti program. Ovaj puta je prevođenje uspjelo, što znamo jer kompajler nije dojavio nikakve greške, a također je stvorio i datoteku sa prevedenom strojnim kodom i nastavkom **`.s`** Ovu datoteku pokrećemo pomoću simulatora **`spim`**:

```
spim -file test.s
```

Dobiti ćete slijedeći ispis.

```
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
The following symbols are undefined:
main

COOL program successfully executed
```

1.12 Ispis

Naš prvi program nije ništa ispisao. Za ispis je potrebno koristiti klasu **`IO`**. Najprije ćemo deklarirati atribut **`i`** klase **`Main`** tipa **`IO`**. Zatim pomoću i možemo vršiti ispis:

```
class Main {
      i:IO;
```

```
        main():Int {
            {
                i.out_string("Hello World!\n");
                1;
            }
        };
};
```

Sada je sadržaj metode **`main()`** blok koji se sastoji od dvije naredbe:

- prva je naredba za ispis
- druga je izraz jedan

Blok mora biti unutar vlastitih vitičastih zagrada (nisu dovoljne postojeće vitičaste zagrade od metode). Svaka naredba bloka treba biti završena sa znakom točke-zareza. Vrijednost bloka u jeziku Cool jednaka je vrijednosti zadnje naredbe bloka. Vrijednost funkcije **`main()`** biti će ujedno i vrijednost bloka.

Prevedite i pokrenite ovaj program. Ovakav program će se uredno prevesti, ali prilikom pokretanja javlja grešku:

```
Dispatch to void.
```

Razlog je što smo objekt **`i`** samo deklarirali, ali nismo alocirali. U jeziku Cool svaki objekt je potrebno alocirati ključnom riječi **`new`** uz koju pišemo klasu objekta koji alociramo.

```
class Main {
        i:IO <- new IO;
        main():Int {
            {
                i.out_string("Hello World!\n");
                1;
            }
        };
};
```

Program se sada može i prevesti i pokrenuti, te ispisuje "Hello World!".

Ovaj program možemo pojednostavniti tako da uklonimo blok i ostavimo samo naredbu za ispis:

```
        class Main {
            i:IO <- new IO;
            main():Int { i.out_string("Hello World!\n") };
        };
```

Prilikom kompajliranja dobiti ćemo slijedeću grešku:

```
        Inferred return type IO of method main does not conform
    to declared return type Int.
```

Razlog je šta metod**a `out_string`** vraća tip **`IO`,** a ne **`Int`.** Promijeniti ćemo tip metode u **`IO`**:

```
class Main {
        i:IO <- new IO;
        main():IO { i.out_string("Hello World!\n") };
};
```

Sada program radi ispravno.

U jeziku Cool svaka klasa je pod-klasa od klase **`Object`**. Zato možemo kao povratni tip metode **`main()`** staviti **`Object`**. Zbog mehanizma nasljeđivanja kompajler neće javiti grešku.

```
class Main {
     i:IO <- new IO;
     main():Object { i.out_string("Hello World!\n") };
};
```

1.13 Privremena deklaracija

Za ispis nije nužno deklarirati i alocirati zaseban atribut u klasi **`Main`**. U jeziku Cool možemo unutar koda privremeno alocirati objekt neke klase i pozvati metodu za njega:

```
class Main {
     main():Object { (new IO).out_string("Hello World!\n") };
};
```

1.14 Nasljeđivanje i ključna riječ self

Također, umjesto privremene deklaracije objekta klase **`IO`**, klasa **`Main`** može naslijediti klasu **`IO`**. Tada metodu **`out_string`** pozivamo pomoću ključne riječi **`self`**. Ona predstavlja sam objekt, kao što je **`this`** u C++ pokazivač na sam objekt.

Nasljeđivanje klase se navodi iza imena klase, pomoću ključne riječi **`inherits`**, iza koje slijedi ime klase od koje se nasljeđuje.

```
class Main inherits IO {
     main():Object { self.out_string("Hello World!\n") };
};
```

Nije nužno eksplicitno pisati **`self`** prilikom poziva metode. Ukoliko je neka metoda pozvana bez navođenja objekta, podrazumijeva se da je pozvana za **`self`**.

```
class Main inherits IO {
     main():Object { out_string("Hello World!\n") };
};
```

1.15 Ulaz

Za čitanje stringa sa tipkovnice koristimo funkciju **`in_string()`** klase **`IO`**.

```
class Main inherits IO {
     main():Object {
          out_string(
               in_string().concat("\n")
          )
```

```
        };
};
```

U ovom programu nakon čitanja ispišemo string. Na učitani string nadovezali smo znak nove linije pomoću funkcije **`concat()`** klase **`String`**. Primjerice, ako upišete string **"jip"** dobit ćete slijedeći ispis:

```
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
The following symbols are undefined:
main

jip
jip
COOL program successfully executed
```

1.16 Konverzija stringa u broj

Ukoliko želimo pročitati cijeli broj, koristimo konverziju iz stringa u cijeli broj. Konverzija se radi pomoću funkcije **`a2i()`** klase **`A2I`**:

```
class Main inherits IO {
      main():Object {
            out_string(
                  (new A2I).i2a(
                        (new A2I).a2i( in_string() )
                        +1
                  ).concat("\n")
            )
      };
};
```

Razmotrimo ovaj kod:

- u kodu najprije učitamo string korištenjem funkcije `in_string()`
- učitani broj potom pretvorimo u cijeli broj pomoću funkcije **`a2i()`** klase **`A2I`** `(new A2I).a2i()`
- potom dodamo 1 sa `+1`
- za ispis koristimo konverziju natrag u string sa `(new A2I).i2a()`
- potom ispišemo konvertirani broj kojemu dodamo prijelaz u novu liniju sa `out_string().concat("\n")`

Kada pokušamo prevesti ovaj program dobiti ćemo nekoliko grešaka, a prva je:

```
'new' used with undefined class A2I.
```

Naime, kôd klase **A2I** se nalazi u drugoj datoteci - **atoi.h**, pa je trebamo priključiti tijekom prevođenja. Da bi preveli više datoteka zajedno, prilikom poziva kompajlera navedemo sve potrebne datoteke navedemo odvojene razmakom:

```
coolc test.cl atoi.cl
```

Kada pokrenete program i upišete recimo broj 12 dobiti ćete ispis:

```
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
The following symbols are undefined:
main

12
13
COOL program successfully executed
```

Ovaj kôd možemo čitkije napisati koristeći članske varijable klase:

```
class Main inherits IO {
      i : Int <- new Int;
      main():Object {{
            i <- (new A2I).a2i( in_string() );
            i <- i + 1;
            out_string(
                  (new A2I).i2a(i).concat("\n")
            );
      }};
};
```

1.17 Zadatak:

Za kraj prve laboratorijske vježbe trebate samostalno napraviti dva programa u jeziku Cool. Sami izaberite dva zadatka od slijedećih ponuđenih:

1. napišite program koji sa tipkovnice učitava dva broja, te ispisuje njihov zbroj na ekran.
2. napišite program koji sa tipkovnice učitava dva broja, te ih ispisuje u obliku kompleksnog broja na ekran. Primjerice ako se učitaju 3 i 5, ispisuje se 3+5i
3. napišite program koji sa tipkovnice učitava dva broja, te ih ispisuje u obliku razlomka na ekran. Primjerice ako se učitaju 3 i 5, ispisuje se 3/5
4. napišite program koji sa tipkovnice učitava tri broja koja predstavljaju dan, mjesec i godinu, te ih ispisuje u obliku datuma na ekran. Primjerice ako se učitaju 3, 5 i 2023, ispisuje se 3.5.2023.
5. napišite program koji sa tipkovnice učitava najprije ime, pa za tim prezime korisnika. Program potom ispisuje prezime i ime korisnika odvojene jednim znakom razmaka.
6. Sami smislite jedan zadatak po uzoru na prethodne, te napišite zadatak i njegovo rješenje.

Rješenje ovoga zadatka trebate predati na kraju laboratorijskih vježbi kao sastavni dio izvještaja sa laboratorijskih vježbi.