



Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Jun 2022.

Autor:

Ilija Kalinić, SW65/2019

Predmet:

Programski prevodioci

Tema projektnog zadatka:

Proširenje miniC klasama

Sadržaj

1	Korišćeni alati	1
2	Evidencija implementiranog dela	1
3	Detalji implementacije	1
4	Ideje za nastavak	1
5	Literatura	2

1 Korišćeni alati

Naziv	Verzija
Bison	3.8.2
Flex	2.6.4
GCC	11.3.0
GNU Make	4.3

2 Evidencija implementiranog dela

U projektu je rađena sintaksna i semantička analiza za definiciju klase, kao i pristupanje članovima klase. U ovoj implementaciji, svi članovi klase su javni i mogu biti ili promenljive koje su već definisane u *micko* kompjajleru ili funkcije, takođe mora postojati posebna funkcija koja inicijalizuje klasu.

3 Detalji implementacije

Sintaksa i implementacija klasa je rađena po uzoru na jezik C++. Svaka klasa definiše novi "korisnički" tip. Klase moraju biti definisane pre funkcije u kojoj se koriste. Implementacija klase je na globalnom nivou (ne postoje lokalne klase, odnosno klase koje su definisane na nivou neke funkcije). Ideja je da klasa počinje ključnom reči i imenom klase, nakon čega postoji telo klase u kojem su definisane promenljive i funkcije klase. Telo klase sadrži promenljive koje se u njoj koriste, funkciju Init() koja inicijalizije vrednosti promenljivih i funkcije klase koje mogu da se koriste. Veliko ograničenje je što unutar tela klase mora postojati Init() funkcija, koja služi za inicijalizaciju same klase. Pri "konstrukciji" klase u nekoj funkciji, navodi se ime klase neke definisane klase kao tip promenljive, zatim treba da se pozove init funkcija klase kako bi se mogli koristiti njene funkcije ili promenljive unutar nje. Po uzoru na C++, promenljivim ili funkcijama neke klase se pristupa preko operatora ". ". Postoji posebna sintaksa za inicijalizaciju klase a to je da se prosto napiše "NekaKlasa.init".

Testni fajlovi su sledeći: *test-class-ok1.mc*, *test-class-ok2.mc*, *test-class-semerr1.mc*.

4 Ideje za nastavak

Prilikom izrade sam naišao na niz problema posebno u delu generisanja koda, iz razloga što sam koristio kompjajler sa jednim prolazom, što za definisanja "korisničkih" tipova nije dovoljno usled potrebe za dinamičnim zauzimanjem memorije pri kreiranju

novih klasa. Da se radilo sa kompjlerom iz dva ili više prolaza, bilo bi mnogo više mesta za popravljanje semantičkih provera, optimizacija i generisanja koda.

5 Literatura

- <https://en.cppreference.com/w/>
- <https://gcc.gnu.org/git.html>
- <https://godbolt.org/>