DOKUMENTACIJA

Programski prevodioci - predmetni zadatak

Osnovni podaci

Broj indeksa	Ime i prezime	Tema
SW69/2019	Ivana Stevanović	Template funkcije i pokazivači

Korišćeni alati

Naziv	Verzija
Flex Bison GCC Compiler	2.6.4. 3.8.2. 11.2.0.

Evidencija implementiranog dela

Prvi zadatak jeste implementacija template funkcija po uzoru na C++. U tom delu je odradjena leksika, sintaksa, semantika, kao i generisanje koda.
Drugi zadatak je implementacija pokazivača, takodje po izoru na C++ programski jezik. Odradjene su leksika, sintaksa i semantika.

Detalji implementacije

- ### Template funkcije

  Omogućeno je definisanje template funkcija, tako da ne smeju postojati ni template ni obične funkcije sa istim imenom. Podržano je definisanje template funkcija (slika 1) sa više parametara, koji moraju biti dinamičkog tipa (u enumeraciji types u defs.h dodat novi tip „T“). Svakom parametru se, kao drugi drugi atribut, pamti indeks funkcije kojoj pripada, kako bi se omogućila upotreba istih naziva parametara izmedju više funkcija.

```
template <typename T> T saberiSveParametre(T a, T b, T c, T d) {

        return a+b+c+d;
}
```

Slika 1. Definisanje template funkcije

Zatim je omogućeno pozivanje template funkcija, kao što je prikazano na slici 2. Tip parametara i povratne vrednosti funkcije se navodi tek pri pozivu, tako što se tip prosledi u <>, nakon imena funkcije. Tada se proverava da li su prosledjeni argumenti i promenljiva u kojoj se smešta rezultat odgovarajućeg tipa. U primeru sa slike 2 neophodno je da i rezultat i sva četiri parametra budu tipa int.

```
rezultat = saberiSveParametre<int>(a, b, c, d);
```

Slika 2. Poziv template funkcije

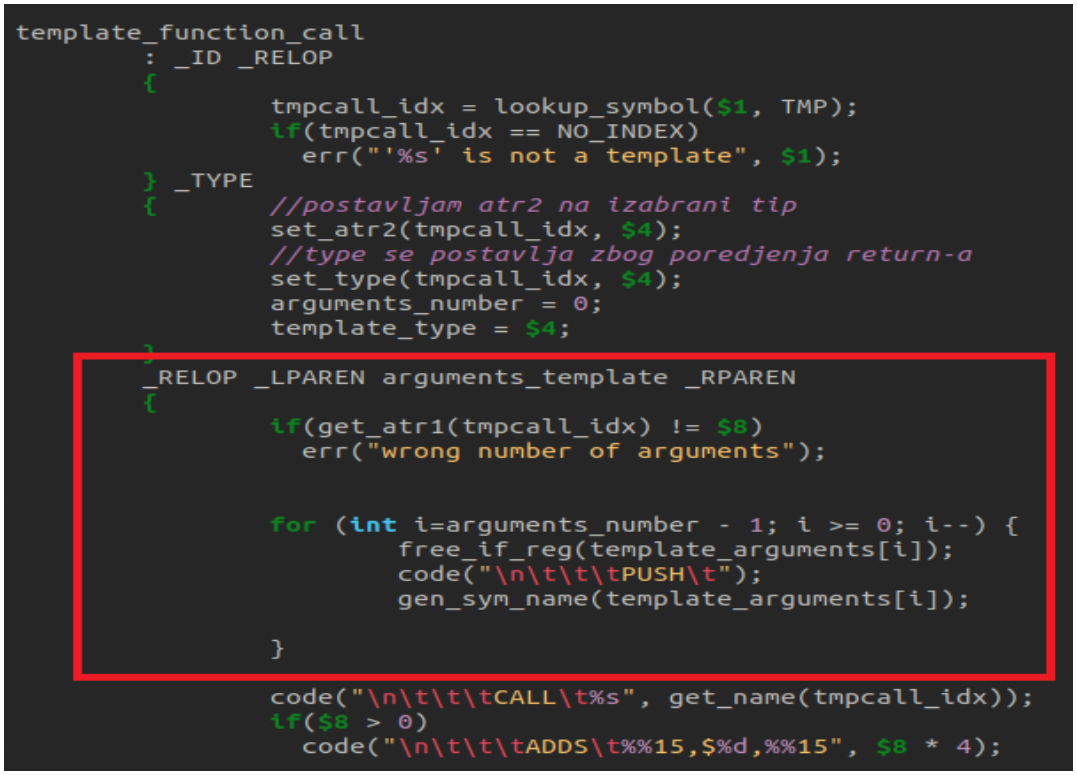
S obzirom da je podržano prosledjivanje više argumenata, bilo je potrebno da se omogući njihovo pravilno čitanje u funkciji. To je odradjeno tako što se najpre prolazi kroz sve argumente i jedan po jedan se ubacuju u niz template_arguments (Slika 3). Nakon toga, kroz pomenuti niz se prolazi otpozadi i argumenti se ubacuju na stek (Slika 4). Tako je omogućeno da se prvi parametar nalazi 8 bajtova više u odnosu na pokazivač stek frejma, drugi +12, treći +16...

```
argument_template
        : num_exp
         {
              if(get_atr2(tmpcall_idx) != get_type($1))
                err("incompatible type for template argument");


              template_arguments[arguments_number]=$1;

              arguments_number = arguments_number + 1;
              $$ = arguments_number;
         }
        ;
```

Slika 3. Dodavanje argumenata template funkcije u niz



Slika 4. Dodavanje argumenata unazad na stek

- **Pokazivači**

  Podržani su pokazivači na promenljive koje su tipa int i unsigned int. Implementirano je dodeljivanje vrednosti pokazivaču. To je ostvareno pomoću operatora &. Nakon operatora se mogu nalaziti isključivo promenljive. Takodje je implementirana i upotreba operatora dereferenciranja (*), odnosno pristup vrednosti. Ukoliko uzmemo da pokazivač p pokazuje na celobrojnu promenljivu x, *p se može pojaviti u bilo kom kontekstu gde bi se moglo pojaviti i x (npr. aritmetičke operacije). S obzirom da su pokazivači promenljive kao i sve druge, moguće je njihovo korišćenje i bez dereferenciranja. Jedan od primera je i dodeljivanje vrednosti jednog pokazivača drugom.

Svi relevantni testovi se nalaze u folderima pokazivači-test i templateFunkcije-test.

Ideje za nastavak

Jedna ideja jeste svakako implementirati generisanje koda kod pokazivača. Zatim, druga ideja jeste da se doradi generisanje koda kod template funkcija tako da se definicija i deklaracija funkcija izgenerišu tek kada se naidje na poziv te funkcije. Razlog tome jeste taj što nam je tip template funkcije zapravo poznat tek pri njenom pozivu, nikako ranije. Ja sam pokušala to da odradim, ali nisam uspela do kraja.

Literatura

1. Predavanje i vežbe
2. https://en.cppreference.com/w/cpp/language/function_template
3. https://www.programiz.com/cpp-programming/function-template
4. https://www.geeksforgeeks.org/pointers-c-examples/