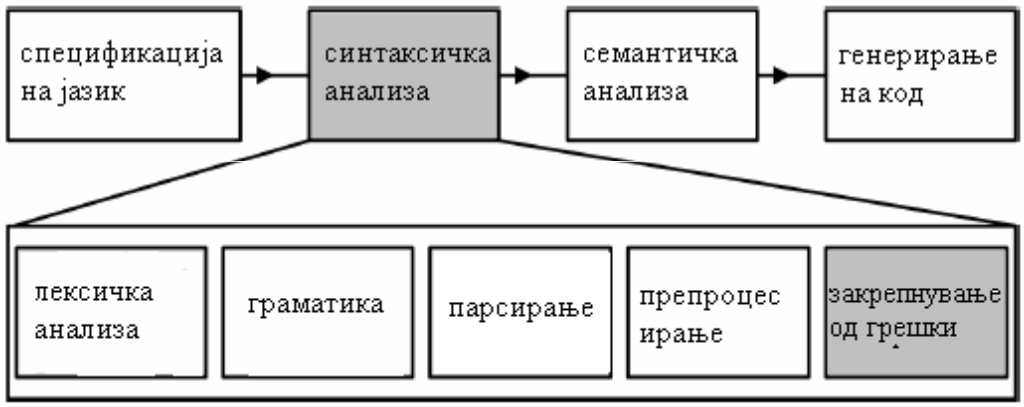
Закрепнување од грешки

Закрепнување од грешки



Вовед

- Скоро секој програм кој ќе се напише содржи некоја грешка
- Бидејќи програмските јазици имаат многу строга граматика, скоро е невозможно да се напише од прва сложена програмска структура која нема ниту една грешка.
- Заради тоа на детектирањето на грешки се обрнува посебно внимание
  - Корисниците често ги користат и треба да се запознаат со нив
  - Битно е пораките за грешка да бидат јасни, точни а со тоа и да можат да се искористат
  - Не треба да се бара корисникот постојано да гледа во прирачник за да знае за каква грешка станува збор, т.е. Пораката што ја дава компајлерот треба јасно да наговестува за каква грешка станува збор.

Работа со грешки

- Сите синтаксички грешки можат да се откријат при парсирањето
- Тие треба да се прикажат во што е можно покорисна смисла
- Сакаме парсерот да открие што е можно повеќе грешки и корисникот по секое компајлирање да поправи што е можно повеќе грешки, се додека програмата не стане целосно синтаксички точна
- Постојат три фази во работата со гршки
  - Откривање
  - Репортирање
  - Закрепнување

Работа со грешки

- Откривањето на грешки се случува во текот на компајлирањето или извршувањето на програмата
  - Compile-time грешките се откриваат во текот на преведувањето-тоа се грешките за кои треба да се грижи компајлерот
  - Runtime грешките се откриваат од страна на оперативниот систем во конјункција со хардверот (како делење со нула)
- Кога една грешка ќе се открие треба да се репортира и до корисникот и до функцијата која ќе ја процесира грешката
  - Корисникот треба да биде информиран за природата на грешката и нејзиното место (бројот на линијата или уште повеќе точниот знак каде се појавила)
- Најтешка задача на компајлерот е да закрепне грешката
  - Закрепнување значи да се врати на позиција во која ќе може нормално да продолжи со парсирањето, па понатамошните грешки да не се јавуваат како резултат на оригиналната грешка

Откривање на грешки

- Compile-time грешките се делат на четири дела:
  - Откривање на лексички грешки
  - Откривање на синтаксички грешки
  - Откривање на компајлерски грешки

Откривање на лексички грешки

- Сканерот репортира лексички грешки кога
- Има влезен знак кој не може да биде прв знак во ниту еден жетон
  - Не го распознава жетонот.
  - Пример за лексичка грешка е незавршен коментар
    - Кога ќе сака да го отфрли, нема да најде каде да заврши коментарот
  - Друг пример е да не се обележи крај на стринг
- Има грешки кои надминуваат меморија
  - Повеќето јазици прифаќаат integer тип со должина од32 бита. Овие грешки не можат да бидат откриени со регуларни изрази (евентуално со регуларни изрази можат да ја пребројат должината, но пример за 000000000000001 ќе рече дека не е добар број). Наместо тоа лексерот превзема акција која се нарекува лексерска акција со која проверува дали жетонот не надминува меморија. Ако е така, тогаш репортира грешка, става 0 на тоа место и продолжува со работа

Откривање на синтаксички грешки

- Парсерот ја користи граматиката за да определи кој жетон се очекува да биде следен
  - Секој нетерминал си има FIRST множество и FOLLOW множество.
  - Откога ќе го добие наредниот терминал од лексичкиот анализатор треба да провери дали се поклопува со FIRST множеството на нетерминалот кој во тој момент се евалуира
    - Ако е во ред-продолжува со работа
    - Инаку, парсерот престанува со работа и се повикува функцијата за процесирање на грешки.
- -

Откривање на семантички грешки

- Семантичките грешки се откриваат во текот на акциите кои се превземаат од страна на компајлерот
- Примери
  - кога се наидува на некоја променлива, таа мора да има каде да зе запише во симболна табела.
  - ако променливата "a" се означува со променливата "b“, тие мора да бидат од ист тип

Откривање на компајлерски грешки

- Попречување од самиот компајлер.
- Точна програма може неточно да се искомпајлира заради “бубачка” во програмата
- Не коже да се направи ништо друго туку да се пријави до тие што го правеле системот
- За да се направи компајлерот со што е можно помалку грешки треба постојано сам да се тестира.

Репортирање на грешки

- Кога ќе открие грешка таа треба да му се јави на корисникот и на функцијата која работи со грешки
- Обично корисникот добива една или повеќе пораки кои ги објаснуваат грешките
- Пораките треба да се придршуваат кон неколку правила
  - Пораките треба да се специфични со природата на грешката, да го дадат местото на грешката што е можно поблизу. Некои компајлери ја даваат само линијата во која е откриена грешка, но други ја даваат точната позиција.
  - Пораката мора да биде напишата во јасна и комплатна реченица, не со шифри, како "error number 33".
  - Не треба да се даваат непотребни информации. На пример ако некоја променлива не е дефинирана, не треба постојано да се јавува, на секое место каде истата се повикува, туку само еднаш.
  - Пораката треба да индицира на природата на откриената грешка. На пример ако се очекува точка запирка, треба тоа да го каже а не само "syntax error" или "missing symbol".
  - Треба да биде јасно дека тоа што е дадено е навистина грешка, а не некоја која компајлерот сам ја генерирал.

Репортирање на грешки-примери

- Грешка: изворниот код содржи integer вредност12345789012345678 90.

  Одговор: оваа грешка може да се сфати како предупредување, ќе се замени со некоја друга вредност, но ќе се јави порака од облик:

  test.i (43): warning: integer value overflow (12345678901234567890). Replaced with 0.
- Грешка: во изворниот код недостасува клучниот збор THEN на место каде што тој се очекува

  Одговор: оваа грешка не може да се сфати како предупредување, затоа што недостасува битен дел од програмата. Мора да се испечати локацијата на грешката и може да се напише порака од облик:

  test.i (54): error: THEN expected after IF condition.

  Да забележиме дека сега не е лесно да се избави од оваа грешка.

Закрепнување од грешки

- Има три начини на кои се врши закрепнување од грешки
  1. Кога ќе се најде грешка компајлерот престанува со работа и не се обидува да најде други грешки
  2. Кога ќе се најде грешка, компајлерот ја репортира и продолжува со парсирање. При тоа не се прави обид да закрепне грешката, па наредната грешка што ќе се најде може да е како резултат на веќе добиената грешка
  3. Кога ќе се најде грешка, компајлерот ја репортира и закрепнува од неа. Наредната грешка што ќе се најде може да не е како резултат на веќе добиената грешка
- Најдобар е последниот пристап, затоа што времето на компајлирање е големо, па добро е да се репортираат повеќе грешки од еднаш.

Синхронизација

- Закрепнувањето од грешки користи така наречени синхронизирачки точки кои парсерот ги бара откога ќе биде откриена некоја грешка
- Синхронизирачка точка е локација во изворниот код од која парсерот безбедно може да продолжи со парсирање без да испечати грешка коа е како резилата на претходната грешка
- За таа цел се користат нножествата
  - FIRST – множество од терминали со кои стрингот почнува
  - FOLLOW – терминали кои можат да се генерираат веднаш после соодветниот нетерминал