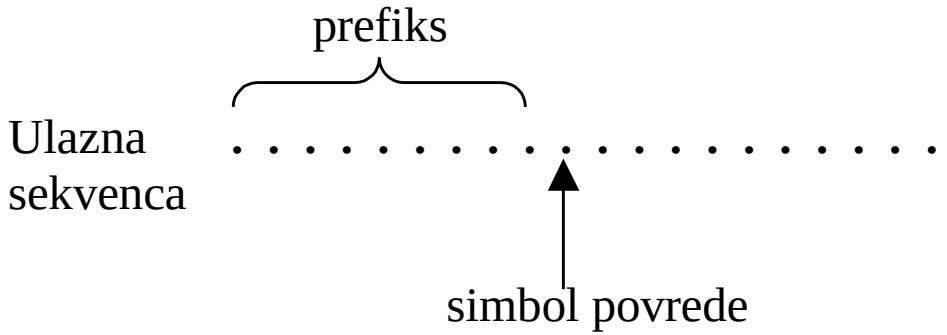
Oporavak od grešaka kod parsera - uvod

- prevođenje se obično ne zaustavlja na prvoj greški u ulaznom programu
- Prihvatljivi odgovori prevodioca pri nailasku na grešku:
  - detekcija greške (izdavanje odgovarajuće izlazne poruke)
  - oporavak od greške – prevodilac pokušava da određenim akcijama učini moguć nastavak sintaksno-semantičke analize programa, da bi se detektovale i druge potencijalne greške. Ne vrši se generisanje izlaznog programa. Akcije obuhvataju modifikaciju parserskog steka i/ili preostalog ulaza.
  - popravak greške – prevodilac pokušava da korisnikov ulaz sa greškom popravi u validan program, uz generisanje prevedenog koda. Naravno, nije garantovano otklanjanje greške u smislu da se postigne tačno ono što je programer želeo. Popravljeni program može nekorektno da se ponaša pri izvršavanju.
- Neprihvatljivi odgovori:
  - prevodilac ne detektuje postojeću grešku u programu
  - prevodilac otkazuje (greška u izvršavanju)
  - prevodilac počinje da se vrti u mrtvoj petlji, stalno prijavljujući istu grešku

Svojstvo prefiksa

- **Simbol povrede** predstavlja onaj simbol ulaznog niza na kome je otkrivena greška.



- Parser ima **svojstvo ispravnog prefiksa** ako za sve neispravne ulazne sekvence niz simbola pre simbola povrede predstavlja prefiks nekog prihvatljivog ulaznog niza.
- Kod takvih parsera možemo modifikacijom preostalog ulaza učiniti ceo ulayni niz prihvatljivim
- Parseri konstruisani metodima koje proučavamo na kursu imaju ovo svojstvo
- Poruka o grešci najčešće se formuliše na osnovu vršnog simbola steka i tekućeg ulaznog simbola, uz navođenje linije i pozicije na liniji koda na kojoj je nastupila greška.
- Potom se vrši oporavak parsera od greške, koji potencijalno može da prouzrokuje nove nepostojeće greške. Kvalitet oporavka ceni se po tome koliko malo nepostojećih grešaka se prijavljuje.

Oporavak od grešaka u SR parsiranju

- izložićemo osnovne (i najčešće primenjivane) varijante "paničnog" oporavka. Ideja je da pri pojavi greške, parser pokušava da izađe iz jezičkog konstrukta u kome je nastala greška, tražeći siguran ulazni simbol (kao što je terminator naredbe ili početna ključna reč sledeće naredbe i slično) od koga može da nastavi normalno parsiranje.
- kompleksniji metodi oporavka i popravka greške (i za LL i za LR tehnike) mogu se naći u knjizi Fisher-a, LeBlanc-a

Jednostavan "panic mode" algoritam:

- Definišemo skup tzv. sigurnih simbola (kao što su terminatori iskaza i konstrukcija ; end --| itd.)
- (1) Kada parser detektuje grešku, sa ulaza se redom uklanjaju simboli dok se ne naiđe na neki od sigurnih simbola.
- (2) Tada se sa steka parsera skidaju simboli sve dok se ne naiđe na neki koji može da potisne sigurni simbol na ulazu na stek.
- (3) Parsiranje se posle toga nastavlja na uobičajen način.

Ovaj algoritam najbolje funkcioniše kod nestrukturiranih jezika, kakvi su Basic ili Fortran; ideja je da se parser izvuče iz pogrešnog konstrukta tako što detektuje njegov kraj i posle normalno nastavi parsiranje.

Poboljšanje za strukturirane jezike je da se pored skupa sigurnih simbola definiše i skup simbola zaglavlja (header), kao što su begin, then i slično.

Poboljšani "panic mode" algoritam:

- (1) Kada parser detektuje grešku, sa ulaza se redom uklanjaju simboli dok se ne naiđe na neki od sigurnih simbola ili simbola zaglavlja.
- (2a) Ako je naišao siguran simbol, tada se sa steka parsera skidaju simboli sve dok se ne naiđe na neki koji može da potisne sigurni simbol na ulazu na stek.
- (2b) Ako je naišao simbol zaglavlja, tada se on potiskuje na stek i posebno markira.
- (3) Parsiranje se posle toga nastavlja na uobičajen način. Kada se redukcijom neke smene ukloni markirani simbol sa steka, prelazi se ponovo na korak (1).

Oporavak od grešaka u yacc/cup parserima

Prijava greške

- pri nastanku sintaksne greške u izgenerisanom parseru, poziva se funkcija **yyerror()** kojoj se prosleđuje opis greške u stringu
- Po povratku iz **yyerror()**, parser može izvršiti oporavak od greške ako u gramatici postoje odgovarajuća pravila. U suprotnom, yyparse() završava rad i vraća nenultu vrednost (neuspešan kraj).

Oporavak od greške

- oporavak od grešaka je pod kontrolom korisnika: postoji poseban gramatički simbol error.
- Simbol error tretira se kao specijalan neterminal, čija se zamena ne definiše smenama, već se razmatra se jedino u slučaju da parser detektuje grešku.
- Tada parser pokušava da odgovarajuću pojavu error u gramatici upari sa pogrešnim delom ulazne sekvence i potom nastavi parsiranje.

Primer gramatike sa error smenom

```
stmt : expr ';'
     | while_stmt ';'
     | if_stmt ';'
     | ...
     |  error ';'
     ;
if_stmt : if '(' expr ')' stmt
```

- Smena sa error znači da ako ulazni niz ne odgovara ni jednoj od "normalnih" smena za `stmt`, parser prelazi u režim oporavka "gutajući" pogrešne ulazne simbole (tj. uparujući ih sa error) sve dok ne naiđe na ";".
- Parser izlazi iz režima oporavka ako može da nastavi o dove tačke normalno parsiranje (SHIFT-uje bar tri tokena na uobičajen način).

YACC Algoritam oporavka od greške

- Po detektovanju greške, parser skida sa steka stanja dok ne dođe do stanja koje ima neposredan oporavak od greške (konfiguraciju X→…•error…).
- Tada parser počinje “preskakanje” ulaznih tokena do trenutka kada dođe neki koji očekuje.
- Tada nastavlja parsiranje ali bez izvršavanja semantičkih akcija. Ako uspešno procesira bar tri tokena sa ulaza, ulaz se resetuje na stanje kada je prestalo preskakanje tokena i parsiranje se ponavlja normalno, ovaj put izvršavajući sve akcije.
- Ako parser nije uspeo uspešno da isprocesira tri tokena, parser odbacuje sledeći token sa ulaza i pokušava da nastavi parsiranje.
- Ako se u pokušajima oporavka dođe do kraja ulaza ili isprazni parserski stek (na primer nije bilo odgovarajućeg stanja koje ima oporavak) parser neuspešno završava rad.

Gde dodati pravila za oporavak od greške u gramatiku?

- Jednostavna i efikasna strategija oporavka je da se preskoči ostatak tekuće linije programa (ili ostatak tekućeg iskaza):

```
stmnt: error ';'  /* on error, skip
                     until ';' is read */
```

- Takođe je korisno upariti zatvorenu zagradu za već parsiranu otvorenu zagradu (ili neki sličan par delimitera). U suprotnom, zatvorena zagrada bi izgledala neuparena i generisala dodatnu suvišnu prijavu greške:

```
primary:  '(' expr ')'
        | '(' error ')'
        ...
        ;
```

- Da bi suzbio suvišne prijave grešaka, parser ne prijavljuje sintaksne greške neposredno posle prve prijave. Tek posle tri ispravno procesirana ulazna tokena, ponovo se omogućava prijavljivanje grešaka.
- Ako se u semantičkoj akcija pravila za oporavak od greške upotrebi yyerrok, parser će odmah omogućiti prijavu novih grešaka (neće čekati da prođu tri uspešna tokena), na primer:

```
primary: '(' error ')' { yyerrok; }
```

- Primetiti da pravila za oporavak od greške mogu sadržati akcije, kao i obična pravila.