Структура на програма в C

Част 1

Иван Георгиев, Христо Иванов, Христо Стефанов

Технологическо училище "Електронни системи",
Технически университет, София

17 февруари 2020 г.

Съдържание

Съдържание

Разпознаване на езици

- Всеки естествен език има набор от правила, които го определят. Този набор наричаме *граматика* на езика.
- Компютърните езици, подобно на естествените езици, също имат граматика.
- *Разпознаването на език* (*language recognition*) е процес, при който поредица от символи се анализира и се определя дали спазва граматиката на езика.
- Този процес най-често се разделя на три етапа:
  - *лексикален анализ* (*lexical analysis*)
  - *синтактичен анализ* (*syntactical/syntax analysis*)
  - *семантичен анализ* (*semantic analysis*)
- Поредицата от символи е валидна за езика, ако по време на трите етапа не са намерени несъответствия с граматиката на езика

Разпознаване на езици

- При компилиране на програма компилаторът прочита символите от файла с кода, написан от програмиста, и се опитва да ги разпознае за съответния език
- Тъй като програмните езици трябва да бъдат разбрани от програма, правилата са по-строги и конкретни, защото не трябва да се допускат двусмислия
- Поради тази строгост на правилата, често компилационните грешки изглеждат неразбираеми за незапознатите с граматиката на езика

```
int main() {
    int
}
// main.c:3:3: error: expected identifier or '(' before '}' token
```

Фрагмент 1: Пример за невалиден код на C и съответната компилационна грешка

Съдържание

Лексикален анализ

- *Лексикалният анализ* е първия етап от разпознаването на език, при който поредицата от символи се превръща в поредица от *лексеми*.
- *Лексемите* са низове от символи с предварително зададено значение в езика
  - Броят, смисълът и начина на разпознаване на лексеми зависи от езика
- С други думи целта на лексикалния анализ е да превърне поредицата от привидно произволни символи до базови понятия, които имат смисъл в езика (т.е. лексеми)
- Ако за някакъв низ от символи от поредицата, не може да бъде намерена съответстваща лексема, то поредицата от символи не е валидна за езика

Лексикален анализ - пример

Примерен лексикален анализ на български език:

- Приемаме, че в българския език има само следните възможни лексеми:
  - *РАЗСТОЯНИЕ* - един или повече символи за празно място
  - *ДУМА* - един или повече символи от българската азбука
  - *ПУНКТУАЦИОНЕН ЗНАК* - един от символите - запетая, тире, точка, точка и запетая, две точки.
- Поредица от символи:
  - *Иван дойде по-късно днес.*
- Поредица от лексеми:
  - *ДУМА, РАЗСТОЯНИЕ, ДУМА, РАЗСТОЯНИЕ, ДУМА, ПУНКТУАЦИОНЕН ЗНАК, ДУМА, РАЗСТОЯНИЕ, ДУМА, ПУНКТУАЦИОНЕН ЗНАК*

Лексикален анализ - пример 2

Пример за невалидна поредица от символи на български език спрямо лексемите от предния пример:

- Поредица от символи:
  - *Иван, номер 4 в класа, дойде по-късно днес.*

Лексикален анализ - пример 2

Пример за невалидна поредица от символи на български език спрямо лексемите от предния пример:

- Поредица от символи:
  - *Иван, номер 4 в класа, дойде по-късно днес.*
- Поредица от лексеми:
  - *ДУМА, ПУНКТУАЦИОНЕН ЗНАК, РАЗСТОЯНИЕ, ДУМА, ГРЕШКА*

Съдържание

Синтактичен анализ

- *Синтактичният анализ* е етап, при който се проверява дали поредицата от лексеми отговаря на *синтактичните правила* на езика
- Пример за синтактично правило в българския език са изреченията.
- Изреченията трябва да съдържат поне една *дума* и да завършват с *пунктуационен знак*.

Синтактичен анализ

- *Синтактичният анализ* е етап, при който се проверява дали поредицата от лексеми отговаря на *синтактичните правила* на езика
- Пример за синтактично правило в българския език са изреченията.
- Изреченията трябва да съдържат поне една *дума* и да завършват с *пунктуационен знак*.
- Без синтактичен анализ следната поредица от символи би била валидна в българския език:

```
.,,-:;.,..--..,.,,-:;.,..--..,.,,-:;.,..--..,
```

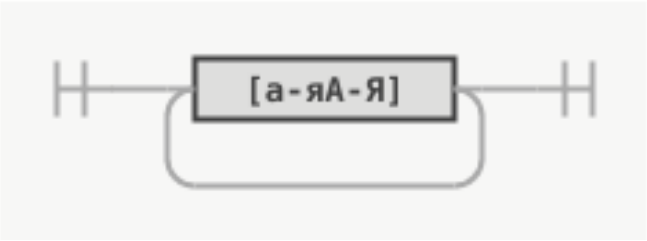
Синтактичен анализ

- Резултата от изпълнението на синтактичния анализ върху поредица от лексеми е поредица от спазени синтактични правила
- При следната поредица от лексеми в българския език:
  - *ДУМА, РАЗСТОЯНИЕ, ДУМА, РАЗСТОЯНИЕ, ДУМА, ПУНКТУАЦИОНЕН ЗНАК, ДУМА, РАЗСТОЯНИЕ, ДУМА, ПУНКТУАЦИОНЕН ЗНАК*

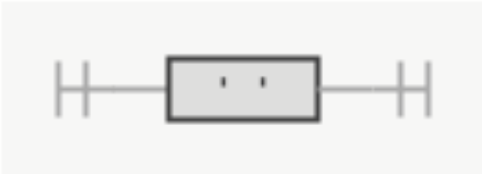
  се получава следната поредица от синтактични правила:
  - *изречение, изречение*

Синтактични диаграми

- *Синтактичните диаграми* служат за графично представяне на правила
- Не всяко правило може да се представи по този начин, но най-често в компютърните езици всички лексикални и синтактични правила могат

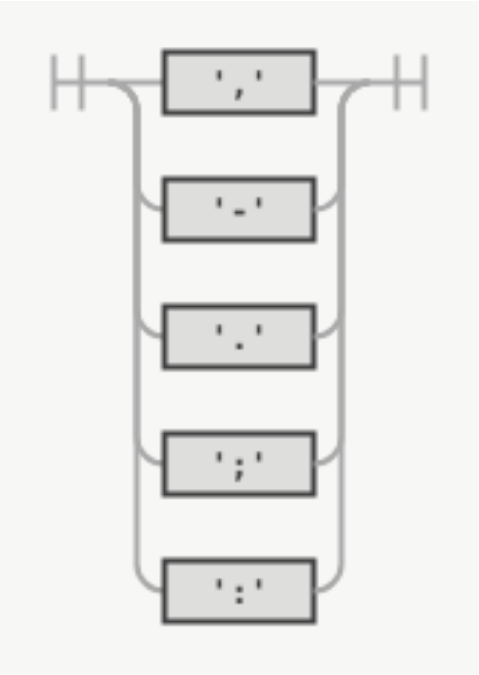


Фигура 1: Синтактична диаграма за дума



Фигура 2: Синтактична диаграма за разстояние

Синтактични диаграми



Фигура 3: Синтактична диаграма за пунктуационен знак

Синтактични диаграми

Фигура 4: Синтактична диаграма за изречение

Фигура 5: Синтактична диаграма за текст

Съдържание

Семантичен анализ

- Не всяка поредица от символи преминала успешно през лексикален и синтактичен анализ е смислена в езика
- *Семантичният анализ* проверява дали поредицата от спазени синтактични правила носят смисъл в конкретния език
- Пример за лексикално и синтактично вярна поредица от символи, но семантично грешна:
  - *Сскфгйаоя флгхкаф дфгхклй аповермнадфи гхклдфгхза.*
- Семантичният анализ е най-сложният етап от разпознаването на език
- По време на изпълнението му се налага да се взимат предвид освен спазените правила, така и конкретните лексеми в самите правила. В зависимост от езика може да се наложи да се разгледат и самите символи, които съставят лексемите.

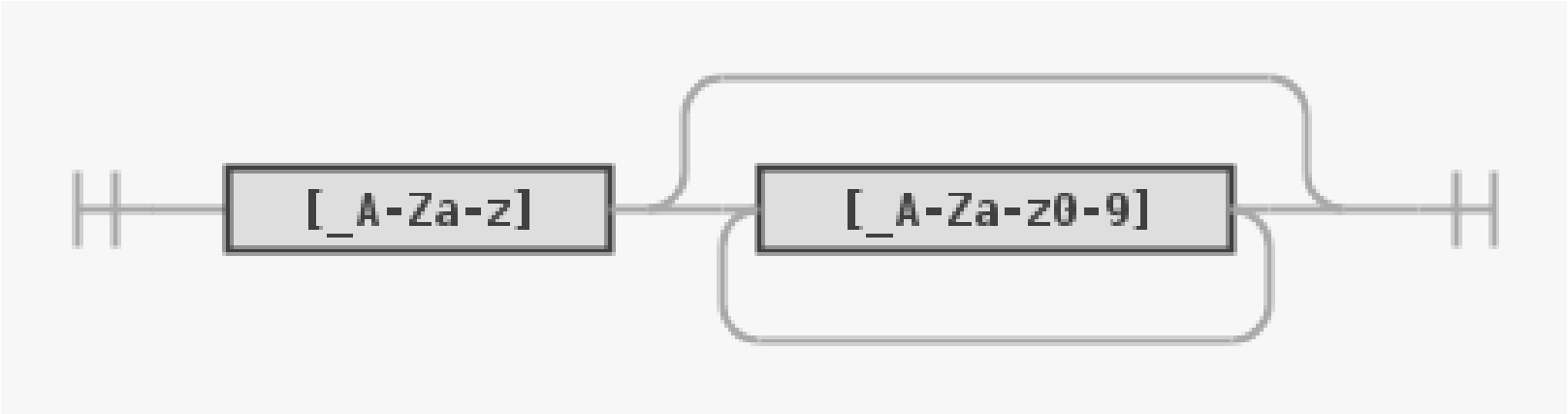
Съдържание

Лексеми в езика C

- Лексемите в езика C са:
  - идентификатор (*identifier*)
  - константа (*constant*)
  - низов литерал (*string literal*)
  - пунктуатор (*punctuator*)
  - коментар (*comment*)
  - празно място (*white space*)

Идентификатори

- *Идентификатор* - някой от символите - '_', 'A' до 'Z', 'a' до 'z' - следван от нула или повече от следните символи - '_', 'A' до 'Z', 'a' до 'z', '0' до '9'
- Примери:
  - a, A, b, _, HELLo, _hello, ___hello, hello1234, hello_world



Фигура 6: Диаграма

Ключови думи

- *Ключовите думи* са лексеми, които отговарят на описанието за идентификатори, но имат различен смисъл в езика от този на идентификаторите.
- Списък на ключовите думи в C:

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Константи

- *Константа* - цяло число в десетичен, шестнадесетичен, осмичен или символен запис[1] или рационално число в десетичен или шестнадесетичен запис
- Примери:
  - Цели числа
    - Десетичен запис - `1, 2, 2019`
    - Шестнадесетичен запис - `0x1f, 0x003, 0xabcdef, 0xABCDEF`
    - Осмичен запис - `0123, 0777`
    - Символен запис - `'j', '0', '\0', '\n', '\152', '\x6A'`
  - Рационални числа
    - Десетичен запис - `1.0, 3.14`
    - Шестнадесетичен запис - `0x1f.3bp0`

[1] Обяснение на следващия слайд

Символен запис

- *Символният запис* на цели числа е символ ограден от единични кавички - ’
- Стойността на числото записано чрез символен запис се определя от номера в ASCII таблицата[2] на оградения символ
- Примери
  - ’a’, ’b’, ’c’, ’!’, ’?’, ’_’ → 97, 98, 99, 33, 63, 95
- За символите от ASCII таблицата, които нямат определен графичен символ (*non-printable characters*, например символ за нов ред, нулев символ) се налага използването на *специални последователности*
- Когато ограденият символ е ’ също се налага използването на специална последователност

[2] http://www.asciitable.com/

Специални последователности от символи

- *Специалните последователности* (*escape sequences*) са последователности от символи, които се интерпретират като един символ от ASCII таблицата.

Последователност	Наименование	Номер в ASCII
\a	alarm or beep	7
\b	backspace	8
\f	form feed	12
\n	new line	10
\r	carriage return	13
\t	tab (horizontal)	9
\v	vertical tab	11
\\	backslash	92
\'	single quote	27
\"	double quote	22
\?	question mark	63
\nnn	octal number	nnn
\xhh	hexadecimal number	hh
\0	null	0

Специални последователности от символи - примери

- Символни записи използващи специални последователности
  - `'\n', '\t'`
  - `'\'', '\\'`
  - `'\047', '\092'`
  - `'\x27', '\x5c'`

Низов литерал

- *Низов литерал* (*string literal*) - низ от символи оградени с двойни кавички - '"'. Може да съдържа специални последователности от символи.
- Примери:
  - "Hello world"
  - "asdf"
  - "First line\nSecond\tline"
  - "String literal with \"quotes\" is also possible"
  - "Single quotes don't need escape sequences in string literals"

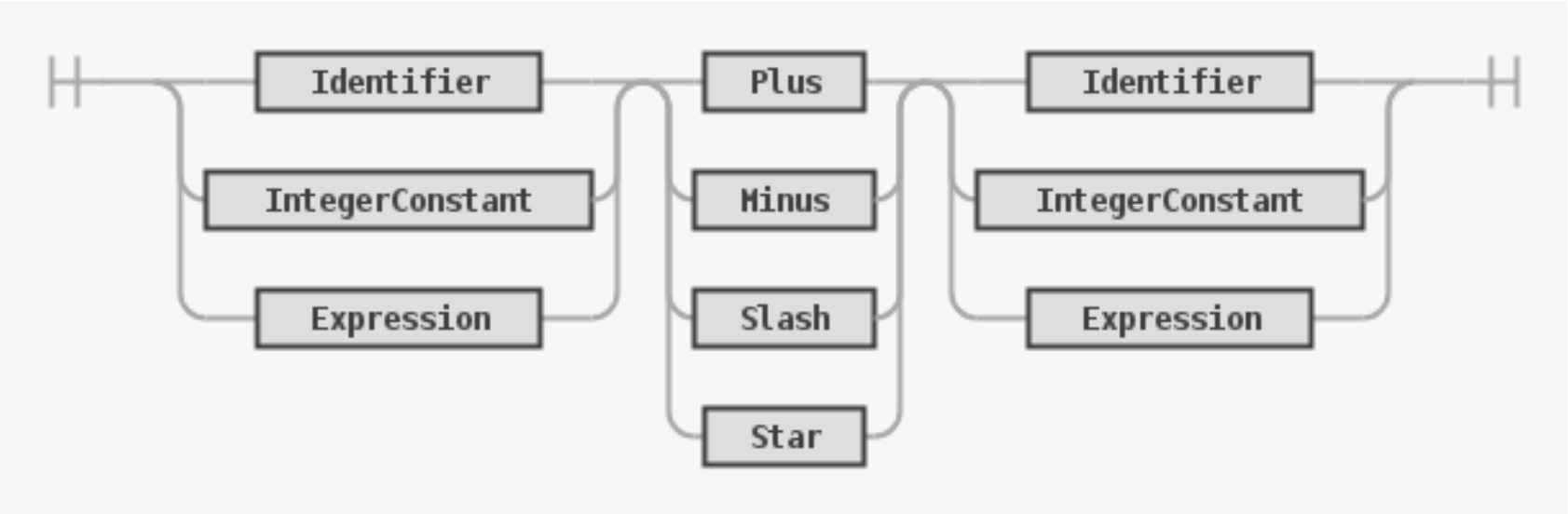
Пунктуатори

- *Пунктуатор* (*punctuator*) - някоя от следните комбинации от пунктуационни символи:

```
[    ]    (    )    {    }    .    ->
&    *    +    -    ~    !
++   -    /    ?    =    ,
%    <<   >>   <    >    <=   >=
:    ;    ...
*=   /=   %=   +=   -=   <<=
==   >>=  !=   &=   ^
|    ^=   &&   ||   |=
```

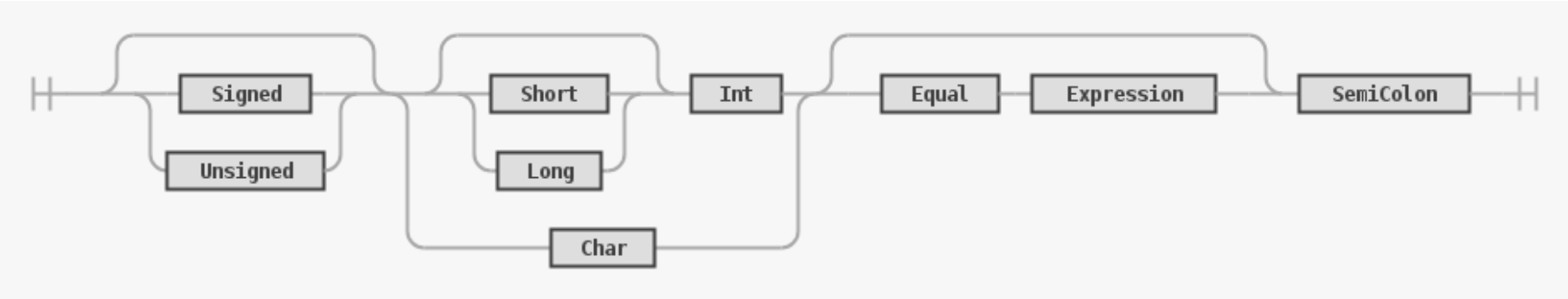
Съдържание

Синтаксис на аритметичен израз



Фигура 7: Непълна диаграма за аритметичен израз в C

Синтаксис на декларация на променлива



Фигура 8: Непълна диаграма за декларация на променлива в C

Съдържание

Семантичен анализ в C

- Пример за семантично правило в C е нуждата да се декларира променлива, преди тя да бъде използвана в израз

```
int main() {
    return a + 5;
}
// test.c: In function ‘main’:
// test.c:2:10: error: ‘a’ undeclared (first use in this function)
```

Фрагмент 2: Пример за лексикално и синтактично верен код, но семантично грешен