

Програмиране

доц. д-р Николай Касъклиев

Учебни материали на имейл
kasakliev@uni-plovdiv.bg

Лекция 1. Езици за програмиране и програми

“Как да накараме компютъра да ни служи?”

Дефиниции

- **Език:**
Система от знаци, използвани за комуникация
- **Език за програмиране (*programming language*)**
 - (1) Система от знаци, използвана от човека при комуникация, която позволява определянето и записването на задачи (по-точно алгоритми) по недвусмислен начин с такава степен на детайлизация, че те да могат да бъдат интерпретирани (изпълнявани) автоматично т.е. от компютър
 - (2) Език за програмиране (ЕП) е система от означения за описание на изчисленията във форма, която може да се прочете от машината и от човека
- **Изчисление (*computation*)**
Всеки процес, извършван от компютъра (математически изчисления, обработка на данни, текстообработка, запазване и извличане на информация ...)
- **Програма (*program, code*)**
Спецификация на изчислението

Програми

Програмата специфицира обработката на данните

За да може компютърът да изпълни определена обработка е необходимо да получи поредица от *разбираеми* команди

За решаване на различните задачи се подават различни *поредици* от команди

Поредицата от команди (наричат се още инструкции) за решаване на определена задача от компютъра се нарича *програма*

Процесът на създаване на програми се нарича *програмиране*

Множеството от команди (инструкции), с помощта на които се записват програмите формира т.н. език за програмиране

Всеки компютър има свой собствен език нарича се машинен език

Програми

Програмата представлява:

ПРОГРАМА=ДАННИ+АЛГОРИТЪМ

- Данни - от изключителна важност са типовете на данните
- Алгоритъм – последователност от операции върху данните решаващи поставената задача

ДАННИ

Данни – тип

- **Числови** (парични суми, брой....)
- **Символни** (имена на стоки, кодове на стоки ...)
- **Логически** (да/не, пр. Пол)

Операции – в зависимост от типа (аритметични за числовите, сравнение на низове за символните...)

Преобразуване на типове – много често прилагана операция от програмиста

АЛГОРИТЪМ

Дефиниции

- Рецепта
- Поредица от елементарни действия описващи изпълнението на дадена задача (изчисление)

Видове

- *Линейни*
- *Разклонени*
- *Циклични*

Елементи на ЕП – примери

Всеки език, включително българския и машинния език има два аспекта – **синтаксис** и **семантика**, които го определят

- “**свдррг**” – не може да бъде дума от българския език, защото нарушава неговите правила за формиране на думи (не е синтактично правилна)
- “**ранижен**” – синтактично правилна, но семантично празна, защото не означава нищо
- “**Безцветни зелени идеи спят разярено**” (Ноам Чомски) – изречение, което е синтактически правилно, но семантично не е

Характеризиращи елементи на ЕП

- Както всеки писмен език, така и езикът за програмиране се основава на азбука:

Азбуката е винаги крайно множество от символи

- **Правилни изречения** на даден език представляват последователности от символите на азбуката, конструирани по т.н. **синтактични правила** на езика (синтаксис, *syntax*)
- Правилните изречения написани на някой от езиците за програмиране се наричат **програми**
- Разбирането, което се влага в така конструираните правилни изречения, т.е. програмите и техните съставни части, се определя от т.н. **семантични правила** на езика (семантика, *semantics*)
- **Взети заедно, азбуката, синтаксисът и семантиката определят езика за програмиране (ЕП)**

Синтаксис и семантика на ЕП

- Строго определен набор от правила на ЕП
- Необходимост:
 - за да бъде програмистът сигурен в правилността и ефективността на програмата, която пише
 - програмата да бъде "разбрана" от компютър и от всеки, който я чете
- *Синтактичните* правила определят как:
 - програмите се изграждат като последователност от символи
 - думите (речникът) на езика могат да се обединяват, за да се получат правилни изречения (да се покаже взаимовръзката им/структурата на изречението)
- *Семантичните* правила:
 - приписват смисъл и значение на комбинациите от думи
 - **т.е. семантиката** е смисълът на градивните елементи на езика и връзката с означените обекти

Синтаксисът определя формата на програмите, а семантиката - техния смисъл

Синтаксис и семантика на ЕП – примери

- *Синтаксис:*
 - изречение, подлог, сказуемо, допълнение, предложен израз, съществително, глагол, прилагателно, и т.н. са синтактични концепции в българския език
 - програма, начало, декларация, тяло, команда, израз, идентификатор и т.н. са синтактични концепции в програмните езици
 - `int a = 2` е синтактично правилно твърдение на C++ (то спазва правилата за образуване на твърдения в C++)
- *Семантика:*
 - `int a = 2;` за програмиста на C++ то означава да се намери ново място в паметта и да се запази в него числото две

Видове езици за програмиране

ЕП най-често се класифицират според:

- областта на приложение
- нивото на абстракция
- програмната парадигма заложена в основата на реализацията им

Видове ЕП според областта на приложение

- **Универсални ЕП** — пригодни за описание на произволни алгоритми ако всяка задача, която може да се реши с компютър, има някакво решение и което може да се изрази с този език
- **Специализирани ЕП** — създадени за специфичен вид обработка

Примери:

- PRG за генериране на доклади
- Logo за графика
- SQL за управление на база данни

Видове ЕП според нивото на абстракция – *I и II поколение*

Разглеждат се четири поколения ЕП, всяко от които е стъпка по посока на писането на програми чрез термини, които представят човешката интерпретация на задачата и резултатите

- ***Първо поколение*** (50-те) – машинни езици

Азбуката, синтаксисът и семантиката на машинните езици се определят изцяло от вида и начина избрани за вътрешното представяне на алгоритмите в компютъра. Всяка команда съответства пряко на едно-единствено действие на машината

- ***Второ поколение*** (до 1954) – езици за асемблиране (комплектоване)
 - позволяват задаване на символни и относителни адреси, както и на потребителски дефинирани команди (*макроси*) в примитивна форма (напр. Add A B вместо 101101010011101001011010) . Програма, наречена *асемблер* превежда всеки символ от програмата на езика за асемблиране и връща символ от машинен език. Като се съберат всички символи от машинния език, асемблерът извежда програма на машинен език

Видове ЕП според нивото на абстракция – *III поколение*

- ***Третото поколение*** (1954 -1970) – ЕП от високо ниво

Абстракцията в техните семантични правила е доста развита, а синтаксисът им е близък до този на естествените език (напр. позволява да пишем изрази от типа $a = 1 + 1$)

Примери: FORTRAN I, Algol 58, Flowmatic (те са базирани на математически изрази), FORTRAN II (въвежда подпрограмите, отделното компилиране), ALGOL 60 (въвежда блоковата структура), COBOL (въвежда структурирането на данните, а обработката на файлове е вградена в самия език), APL (за научни изчисления), LISP (функционално програмиране за изкуствен интелект), PL/I (на IBM), Pascal (наследник на ALGOL 60), SIMULA (класове, абстракции от данни, обектно-ориентирано програмиране)

Видове ЕП според нивото на абстракция – *IV* поколение

- **Четвъртото поколение** (1970-80) – ЕП, които се характеризират с промяна в семантиката

Позволяват на програмистите да изразяват идеите си за нещата по начин естествен за човека, чрез понятия, които не съответстват на никоя компютрна концепция

Примери: Smalltalk (ООП), С (системно програмиране), Prolog (изкуствен интелект), Modula-2 (модули: капсулиране, скриване на информация, абстрактни типове данни, базиран на *обекти*), Ada (пакети: капсулиране, скриване на информация, *абстрактни типове данни*, поддръжка на паралелизъм), C++, Object-Pascal, Modula-3 (добавят *обектно-ориентирани* възможности към съществуващите езици).

Най-мощните от съществуващите езици за програмиране са Visual BASIC, Borland Delphi, Visual C++ (езици за лесна разработка на ГПИ), Perl, Awk, Sed, Tk-Tcl (скриптови езици), Java (уеб-програмиране)

Видове ЕП според парадигмата на програмиране – *Процедурно (императивно) програмиране*

Това е стандартният (традиционен) програмен стил, в който програмите се раздробяват на отделни стъпки (части, наречени *процедури*), чрез които се описват изчисленията

- **Характеристики** на процедурните ЕП:
 - базирани на фон Ноймановата архитектура
 - изпълнението на програмите е последователно
 - променливите представят определени места в паметта
 - за промяна стойността на променливите се използва операцията присвояване
- **Примери:** Pascal, C, Ada

Видове ЕП според парадигмата на програмиране – *Функционално (апликативно) програмиране*

Функционалният стил на програмиране се основава на теорията на математическите функции. Функциите са основните градивни блокове на програмата. Изпълнението на програмата представлява изчисляване на стойности чрез *изрази* и *функции*. Те могат свободно да бъдат предавани като параметри, както и да бъдат конструирани и връщани като параметри - резултат на други функции

- **Характеристики** на функционалните ЕП:
 - функциите се прилагат върху стойности, както и върху други функции за създаването на по-сложни функции
 - предаване на параметри
 - липсва понятието променлива
 - няма цикли, но широко се използва рекурсията
- **Примери:** Lisp, Miranda, ML, Haskell

Примерна програма на Miranda

```
output = title ++ captions ++ concat (map line [1..20])
```

```
title = cjustify 60 "A TABLE OF POWERS" ++ "\n\n"
```

```
captions = format "N" ++ concat (map caption [2..5]) ++ "\n"
```

```
caption i = format ("N^" ++ shownum i)
```

```
format = rjustify 12
```

```
line n = concat [format (show(n^i)) | i<-[1..5]] ++ "\n"
```

Видове ЕП според парадигмата на програмиране – *Логическо (декларативно) програмиране*

Този стил изисква **декларативно описание на задачата**, вместо задаване на отделните стъпки на алгоритъма на задачата. Декларативните програми са по-близки до спецификацията, отколкото традиционното програмиране

- **Характеристики** на логическите ЕП:
 - те са базирани на формалната логика
 - няма цикли
 - програмите се състоят от аксиоми (факти), правила за правене на заключения и цел
 - програмата описва кое е вярно по отношение на очаквания резултат
 - изпълнението на програмата представлява правене на извод
- **Примери:** Prolog и Clips (езици базирани на правила)

Примерна програма на Prolog

```
get_answer( A ) :-
```

```
    ratom( X ), name( X, String ),
```

```
    valid_resp( String, A ), !.
```

```
valid_resp( [H|T], A ) :- type_ans( H, A ).
```

```
type_ans( X, A ) :- ([X] = "y"; [X] = "Y"), A = yes.
```

```
type_ans( X, A ) :- ([X] = "n"; [X] = "N"), A = no.
```

```
type_ans( X, A ) :- ([X] = "w"; [X] = "W"), A = why.
```

```
?-print("\nYou can start animal by typing "help.<CR>"\n' ).
```

```
valid_resp( [], A ) :-
```

```
    print("\nPlease try to give me a yes or no answer.'),
```

```
    get_answer( A ), !.
```

```
valid_resp( [H|T], A ) :- valid_resp( T, A ).
```

Видове ЕП според парадигмата на програмиране – *Обектно-ориентирано програмиране*

При обектно-ориентирания стил на програмиране се набляга на **дефинирането на класове от обекти**. Нужните екземпляри (обекти) от съответен клас се създават по време на изпълнението на програмата

- **Характеристики на обектно-ориентирания ЕП:**
 - обектът е съвкупност от клетки в паметта и операции, които променят стойностите в тях
 - най-простият обект е променливата
 - изчислението е взаимодействие и комуникация между обектите
 - всеки обект се държи като компютър – със собствена памет и операции
 - класът е група от обекти с еднакви свойства
 - обектите се създават като екземпляри на класовете
- **Примери:** Smalltalk, Eiffel, Java, C++, C#

ЕП и парадигми на програмиране

<i>Процедурни/ императивни</i>	<i>Декларативни</i>		<i>Обектно- ориентирани</i>
	<i>Функционални/ апликативни</i>	<i>Логически</i>	
Machine code Assembly FORTRAN COBOL ALGOL PL/I Pascal Modula-2 BASIC C C++, Java, Modula-3, Oberon	Haskell ML LISP Miranda Logo APL	PROLOG и производни	Smalltalk Simula C++ Objective-C Oberon Modula-3 Java



Ще бъдат разгледани характеристиките и основните елементи на универсалните процедурни ЕП от високо ниво

Методи за описанието на семантиката на ЕП

Ще представим два начина на описание на семантиката на ЕП:

- учебници-справочници за езика (с помощта на някакъв човешки език)
- формално дефиниране:
 - аксиоматична семантика (определят смисъла на команда в програма, като описва ефекта върху състоянието на програмата. Твърденията са логически - предикати с променливи, където променливите определят състоянието на програмата.)
 - операционна семантика (формализира, като описва значението като поредица от стъпки - изчисления)
 - денотационна семантика (още се нарича математическа формализира значението на ЕП с математически обекти)

При изучаване на даден ЕП семантичните му правила често се описват неформално, с изречения на естествен език

Методи за описанието на синтаксиса на ЕП

Синтактични правила на един ЕП са сравнително малко и точно определени, ето защо е удобно те да се описват чрез строг и *формален език*

Най-известните методи за специфициране на синтаксиса на един програмен език са:

- форми на Бекус-Наур
- синтактични диаграми

Двата метода

- в основата си са еднакви, различават се само по възприетите начините за означение при задаване на синтактичните правила
- използват понятията:
 - **терминален символ** – дума от речника на езика
 - **нетерминален символ** – по-сложна синтактична конструкция, съставена от терминални символи и други синтактични конструкции. Нетерминалните символи трябва да бъдат заместени от терминални (според правилата), за да се получи правилна конструкция на ЕП

Описание на синтаксиса на ЕП чрез форми на Бекус-Наур (БНФ)

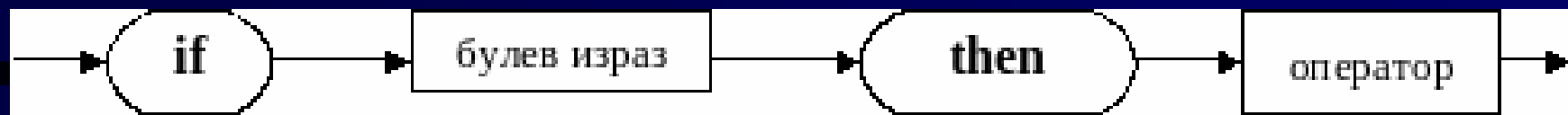
Следният пример описва синтаксиса на реалните числа с цяла част, десетична точка и дробна част в езика Pascal.



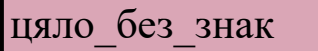

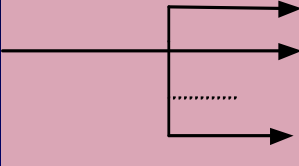
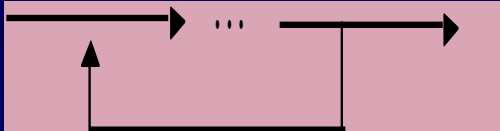
Пример.

$\langle \text{реално-число} \rangle ::= \langle \text{последователност} \rangle . \langle \text{последователност} \rangle$
 $\langle \text{последователност} \rangle ::= \langle \text{цифра} \rangle | \langle \text{цифра} \rangle \langle \text{последователност} \rangle$
 $\langle \text{цифра} \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

- Създателите са Джон Бекус и Питър Наур. За първи път БНФ се използва при описанието на синтаксиса на Algol 60
- *Деривационният символ* от всяка продукция се записва ::=
- *Нетерминални* символи – думи или фрази, записани в ъглови скоби (< ... >)
- *Терминални* символи се записват в кавички (“...”)
- Други *метасимволи* са:
 - { } – нула или повече повторения
 - [] – част от синтаксиса, която не е задължителна
 - Символът | е логическо “или”

Описание на синтаксиса на ЕП чрез синтактични диаграми

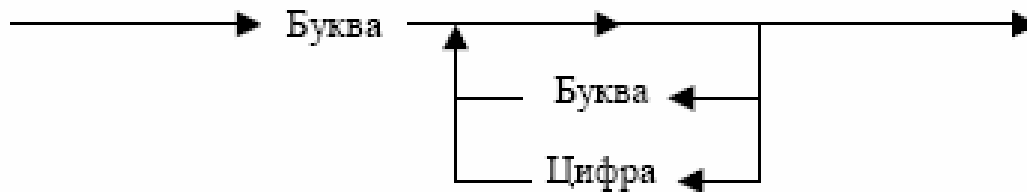


- Графичен език за описанието на синтакс на ЕП
- Над входната стрелка се записва името на синтактичната категория, която се определя с дадената синтактична диаграма
- **Нетерминални** символи — луми или фрази, записани в правоъг.блок ()
Например  или 
- **Терминални** символи се записват в овали. Напр. if, then
-  — стрелката свързва отделните елементи на диаграмата и показва посоката на обхождане. Всяка диаграма има по една входна и изходна стрелка
-  — разклонение
-  — цикъл

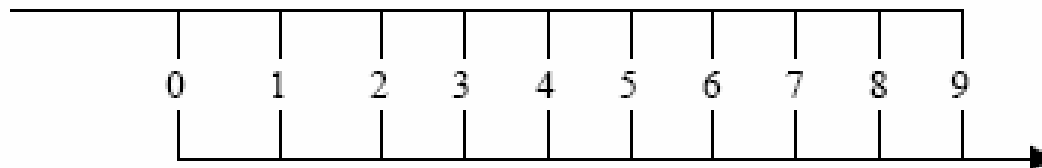
Описание на синтаксиса на ЕП чрез синтактични диаграми

Синтаксис на идентификатор

Идентификатор::



Цифра::



Буква::

