***Езици за програмиране и програми***

Езикът за програмиране е система от означение за описание на изчисления във форма, която може да се прочете от машина и от човек.  
***Изчислението*** е всеки процес, извършван от компютъра. /матем. изчисления, текстообработка и др./  
***Програмата*** е поредица от команди (инструкции) за решаване на определена задача от компютър.  
***Програмирането*** е процесът на създаване на програми.  
**Езикът за програмиране** е множеството от команди (инструкции), с които се записват програмите.  
Програма = Данни + Алгоритъм  
**Данните** биват 3 типа – *числови, символни и логически*.  
**Операциите** зависят от типа на данните /аритметични за числовите, сравнение на низове за символните…/. Програмиста често преобразува типове.  
**Алгоритъма** е поредица от елементарни действия, описващи изпълнението на дадена задача (изчисление). Има 3 типа алгоритми – *линейни, разклонени и циклични.*Езикът за програмиране се определят от азбуката, синтаксисът и семантиката взети заедно.  
Синтаксисът и семантиката са необходими на езика за програмиране за да бъде програмиста сигурен в правилността и ефективността на програмата, която пише и за да може програмата да бъде разбрана от компютъра и от всеки, който я чете.  
Синтаксисът определя формата на програмите, а семантиката – техния смисъл.  
Най-често езиците за програмиране се класифицират според ***областта на приложение***, ***нивото на абстракция*** и ~~програмната парадигма, заложена в основата на реализацията им~~.

Според областта на приложение биват **универсални** и **специализирани**.  
**Универсалните** са пригодени за описание на произволни алгоритми, с тяхна помощ се решават широк кръг от задачи.  
**Специализираните** са създадени за специфичен вид обработка, например за графика /Logo/ или управление на база данни /SQL/ .

**Според нивото на абстракция** програмните езици са четири поколения, всяко следващо от които е по-близо към писането на програми чрез термини, които представят човешката интерпретация на задачата и резултатите.  
***1-во поколение***: всяка команда съответства пряко на едно единствено действие на машината.  
***2-ро поколение***: позволяват задаване на символни и относителни адреси, какти и на потребителски дефинирани команди (макроси).  
***3-то поколение***: семантичните правила са развити, а синтаксисът им е близък до този на естествения език.  
***4-ро поколение***: характеризират се с промяна в семантиката, позволяват на програмиста да изразява идеите си за нещата по начин, естествен за човека, чрез понятия, които не съответстват на никоя компютърна концепция.

***Езици за програмиране и тяхната реализация***

За да бъде използван един новосъздаден език за програмиране от високо ниво той трябва да се реализира чрез някой от преди това съществуващите и реализирани езици, т.е. трябва да бъде реализиран на машинен език, който е материализиран чрез хардуера на машината.

За да бъде изпълняван един език за програмиране от високо ниво той трябва да може да се превежда на машинен език, т.е. написаните от програмиста програми в текстов вид трябва да се превеждат във вид, който може да се изпълни от компютъра.

***Транслацията (translation)*** е процесът на превеждане на текстовия код на програмата в изпълним (двоичен) код.  
***Транслаторът (translator)*** е инструментът, който превежда текстовия код на програмата в изпълним код.

Т.е. транслаторът е инструментът, който превежда кода в машинен език, а самия процес на превеждане се нарича транслация.

Има два вида транслатори – ***компилатори*** и ***интерпретатори***.  
***Компилатора (compiler)*** превежда програмата от текстовия код на езика в програма на друг език за програмиране. По-трудни са за реализиране, по-бавно компилират, но пък изпълняват програмата по-бързо и по-ефективно при многократно изпълнение. Използват се за комерсиални софтуери.  
***Интерпретатора (interpreter)*** чете текста на програмата от езика и я изпълнява директно твърдение след твърдение. По-лесни са за реализиране, по-бързо компилират, но изпълняват по-бавно програмата. По-бързи са при еднократно изпълнение. Използват се аз експериментални разработки, напълно нови езици за програмиране.

**Процесът програмиране включва следните действия**: ***написване***, ***превеждане***, ***изпълнени***е, ***проверка и отстраняване на грешки, модифициране и разширяване и допълнение*** на функционалните възможности на програмата.

Възможните ***видове грешки*** при създаването на програма могат да бъдат 3 вида:  
**Синтактични**  
**Грешки по време на изпълнението** – когато конструкциите са синтактично правилни, но безсмислени.  
**Логически** – когато е синтактично и смислово правилна, но е грешна спрямо действията, които програмата трябва да изпълнява.

***Средата за програмиране*** е съвкупността от програмни средства и инструменти за разработка на програми на някакъв програмен език. Тя осигурява цялостна поддръжка на програмните дейности, улеснява всички етапи от разработването, осигурява създаването на високо-качествени програми.  
Могат да включват ***редактори***, ***дебъгери***, ***компилатори***, ***свързващи редактори*** и др.

**Програмните редактори** служат за написване на програмата и създаване на файловете, от които се състои тя. Програмните редактори *улесняват кодирането*, *подпомагат структорирането на текста* и *предупреждават за синтактични грешки*.

**Транслаторите** превеждат програмата, написана с помощта на редактора, на език, разбираем за машината и я записва във файл. Те подпомагат откриването и премахването на синтактични грешки и оптимизират. Файлът, създаден от транслатора, се нарича обектен код.  
 ***Обектният код*** е предимно машинен код. Той притежава повечето от изграждащите блокове на програмния код, освен системния код, който се намира в биоблиотеките.

**Свързващите редактори (Linker)** създават изпълнимия файл като свързват обектния код със съответните библиотеки.

Зареждащата програма (Loader) зарежда изпълнима програма, създадена от компилатора в паметта на компютъра с цел тя да бъде изпълнена. Зареждащата програма е част от операционната система. След зареждането на програмата операционната система казва на процесора да започне изпълнението на последователността от инструкции от мястото в паметта, където сега се намира програмната.

Дебъгерите (Debugger) подпомагат откриването и премахването на грешките по време на изпълнението на програмата. Самият процес се нарича дебъгване. (bug==грешка)

**C++** съчетава принципите на две програмни парадигми – процедурната и обектно-ориентираната.  
Създаден е от д-р. Строуструп през 80-те години.

**Основни лексикални елементи на програмите**

Лексикалните елементи (token /тукен/ - символ) са основните градивни елементи на програмите, които са значещи за компилатора:

***Име, идентификатор (identifier)*** – последователност от символи, което служи за идентификация на алгоритми, подалгоритми, елементи на данните и др.  
***Ключова дума (keyword)*** – предварително дефинирани имена, които имат предварително зададено предназначение и не могат да бъдат използвани за други цели.  
***Константа (constant)*** – стойност, която не се променя по време на изпълнение на програмата, а само се използва.  
***Операция (operator)*** – задава намирането на някаква стойност по дадени операнди.  
***Коментар*** – това е текст, който се игнорира от транслатора, но е полезен за програмиста.  
***Препинателен знак (punctuator)*** – а бе ти сериозно ли?  
Лексикалните елементи на програмите се отделят с разделители като празен интервал, край на ред, коментар и др.

**Основни елементи на езиците за програмиране от гледна точка на синтаксиса и семантиката:**

**Декларирането и дефинирането на данните** служи за определянето на входните и изходните данни, на техния тип и структура, както и свързаните с тях изисквания за памет.

**Деклариране и дефиниране на обработка** – определя правилата за обработка на данните и последователността на тяхното прилагане чрез оператори.

Декларацията казва на компилатора, че съответния програмен елемент съществува.  
Дефиницията определя кой код или данни идентифицира декларираното име.  
Името трябва да бъде декларирано преди да бъде използвано.  
Всеки елемент данни в програмата е или константа или променлива.

**Абстракциите** в езиците за програмиране се използват за да пресъздадат човешкото разбиране за данните и тяхната обработка, отделяйки го от компютърното разбиране. Те са нужни за да може програмистите да описват програмите по начин, който има смисъл за самите тях.

**Видове абстракции в ЕП:**

**Абстракции на данните** – служат за изразяване на свойствата на данните и тяхната функционалност.  
1.***Базови абстракции на данни*** – представяне на данните на базата на техния тип. Те се наричат **прости типове данни**.  
2.***Структурирани абстракции на данни*** – абстракции на съвкупности от данни. Те се наричат **структурни типове данни**.

**Абстракции за контрол на управлението (управляващи конструкции)** – предназначени са да управляват последователността на изпълнението.  
1.**Базови абстракции за контрол** – оператори, които комбинират няколко машинни инструкции. Те се наричат ***прости оператори***.  
2.**Структурирани абстракции за контрол** – оператори. Които разделят програмата на групи от инструкции и управляват тяхното изпълнение. Те се наричат ***структурни оператори***.

**Абстракция на данните. Базови абстракции на данните.**

**Данните** са общо понятие за всичко това, с което оперират изчислителните машини.  
**Модел на данните:  
 1.абстрактен модел  
 2.знаков модел  
 3.физически модел**

**Абстрактният модел на данните** определя присъщите им свойства и функционалност, като игнорира детайлите на тяхното физическо представяне.  
Абстракцията на данни е разделянето на логическите свойства на данните от тяхното конкретно приложение.

Според своята **структурност видовете абстрактни модели** биват ***прости*** (базови абстракции на данните) и ***съставни*** (структурирани абстракции на данните).