

Урок №1

Програмиране на езика

С

## Съдържание

1. [Предварителни разсъждения 3](#_bookmark0)
2. [Инсталиране на Microsoft Visual Studio 2013 9](#_bookmark1)
3. [Първи проект 18](#_bookmark2)
4. [Извеждане на данни 31](#_bookmark3)
5. [Типове данни 37](#_bookmark4)
6. [Променливи и константи 41](#_bookmark5)
7. [Въвеждане на данни 50](#_bookmark6)
8. [Литерали 54](#_bookmark7)
9. [Домашно задание 56](#_bookmark8)

# Предварителни

разсъждения

## Встъпително слово.

Добре дошли в света на програмирането! Какво е всъщност програмирането? Сигурно всеки от вас някога е чувал тази дума. Префразирайки известен цитат, можем да кажем така: «Програмист — това звучи гордо!» И това действително е така. Ако Вие, сърфирайки из Internet пространството, сте се натъквали на сайтове с предложения за работа, то сигурно сте обърнали внимание на размера на заплащането, което се предлага на програмистите. Естествено, веднага възниква въпрос: защо е толкова високо нивото на заплащане на труда на програмиста? Всичко е свързано със законите на пазара: има търсене на програмисти, но към даден момент действително добрите специалисти не са много. Разбира се, ако програмист можеше да се става за кратко време, научавайки няколко думи, професията едва ли щеше да е толкова популярна. Но не се отчайвайте! Ние, Вашите преподаватели, ще се постараем да направим всичко, което зависи от нас, не само за да Ви разкажем как се програмира, а и да Ви научим да живеете, програмирайки. Обаче, нашата с Вас съвместна работа трябва да бъде взаимна. Преподавателят, колкото и да се старае, не би могъл да обучи студента, ако последният не иска това. Работа, само постоянна работа върху себе си ще Ви заведе до върховете на професионализма.

Искрено се надяваме, че Вие ще бъдете запленени от вълшебния свят на програмирането!!!

Преди да пристъпим към изучаването на която и да е наука, винаги трябва най-напред да изясним откъде се е появила тя, как се е развивала, какви са нейните корени и значението й в историята.

## Исторически факти

Английският математик от XIX век Шенкс отдал повече от 20 години от живота си за да изчисли числото Пи с точност 707 значещи цифри след запетаята. Този резултат придоби славата на рекорд в изчисленията на XIX век. Обаче впоследствие било открито, че Шенкс е сгрешил в520-я знак, поради което всички следващи значещи цифри били изчислени погрешно.

През 1804 година френският изобретател Жозеф Мари Жакар създал «програмно-управляем» тъкачен стан. За управление на стана се използвали перфокарти, свързани помежду си във вид на лента. Дървените шипове на «четящото устройство на стана според разположението на перфорациите в картата определяли, кои конци следва да се повдигнат, а кои да се пуснат за да се получи търсената шарка. през 1890 година в САЩ изобретателят Херман Холерит разработил електромеханично сметачна машина-табулатор, управлявана с помощта на перфокарти. Тя била използвана за съставяне на таблици с резултатите от преброяването на населението на САЩ. Основаната от Холерит фирма за производство на табулатори впоследствие се превърнала в корпорация

International Business Machines (IBM).

През 1936 година 25-годишният студент в университета в Кембридж англичанинът Алън Тюринг публиковал статия «За изчисляемите числа», в която разглеждал хипотетично устройство («машината на Тюринг»), универсално за решаване на всяка разрешима

математическа или логическа задача, — праобраз на програмируемия компютър.

През 1941 година немският инженер Конрад Цузе построил действащ компютър Z3, в който се използвала двоична бройна система. Програмите се записвали на перфолента.

През 1945 година във висшето техническо училище към университета в Пенсилвания (САЩ) физикът Джон Мокли и инженерът Проспер Екърт построили завършена електронна машина «ENIAC». За да се зададе програма било необходимо ръчно да се нагласят хиляди превключватели и да се вкарат стотици щепсели в гнездата на контактния панел. На 1 юни 1945 година бил разпространен отчет на американския математик от унгарски произход Джона фон Нейман «Предварителен отчет за машината Едвак», съдържащ концепция за съхраняване на команди за компютъра в неговата собствена вътрешна памет.

На 21 юни 1948 година в университета Манчестър (Великобритания) на машината «Марк-1» е изпълнена първата в света съхранявана в паметта на машината програма — търсене на най-големия съмножител на зададено число.

През 1949 година под ръководството на Морис Уилкс е създаден компютърът «EDSAC» (Electronic Delay Storage Automatic Computer). Създателите на EDSAC въвели система за мнемонични обозначения, където всяка машинна команда била представена с една главна буква, и автоматизирали настройките на подпрограмите към определено място в паметта. Морис Уилкс нарекъл мнемоничната схема и библиотеката подпрограми «събирателна система» (assembly system) — оттук е понятието «асемблер».

През 1949 година във Филаделфия (САЩ) под ръководството на Джон Мокли е създаден «Краткият код» — първият примитивен интерпретатор на езика на програмиране.

През 1951 година във фирмата Remington Rand американската програмистка Грейс Хопър разработила първата транслираща програма. Хопър я нарекла компилатор (compiler).

През 1957 година на 20-я етаж на щаб-квартирата на IBM на Медисън авеню в Ню-Йорк се родил езикът Фортран (FORmula TRANslation — транслиране на формули). Ръководил групата разработчици 30-годишният математик Джон Бекъс. Фортран е първият от «истинските» езици на високо ниво.

През 1963 година бил създаден езикът на програмиране Бейсик. Негови създатели са Джон Кемени и Томас Курц от Дартмут Колежа. Под тяхно ръководството езикът бил реализиран от група студенти в колежа. Най-първият диалект на езика се наричал Dartmouth BASIC. От 1958–1968 години се разработвал и усъвършенствал език на програмиране с наименование Алгол – съчетание на «алгоритмичен» и «език» (**algo**rithmic **l**anguage). За разлика от Фортрана,

използван предимно в САЩ и Канада, Алгол бил широко разпространен в Европа и СССР. Езикът бил създаден от международен комитет, в състава на който влизали европейски и американски учени — Джон Бекъс, Питър Наур, Уоли Фойрцойг, Никлаус Вирт.



През 1970 Никлаус Вирт създал език на програмиране, който нарекъл в чест на френския физика и математика Блез Паскал. Паскал бил планиран от Вирт като език за обучение на процедурно програмиране. През 1972 година 31-годишният специалист по системно програмиране от фирма Bell Labs Денис Ричи разработил езика на програмиране Си.

Първото описание на езика било дадено в книгата на Б. Керниган и Д. Ричи „Програмният език С“, преведена на български език. Дълго време това описание било стандарт, обаче отделни моменти в него достартали нееднозначно тълкуване, което пораждало множество различни трактовки на езика С. За преодоляване на тази ситуация към Американския национален институт за стандартизация



(АNSI) бил създаден комитет за стандартизация на езика С. През 1983 година бил утвърден стандарт за езика С, получил наименование ANSI С.

В началото на 80-те години в същата Веll Lаbоratory Бьорн Страустроп в процеса на допълване и разширяване на езика С фактически създава нов език, наречен «С с класове». През 1983 година това название било променено на С++.

На 23 май 1995 година компанията Sun Microsystems представя нов език на програмиране с название Oak. Език бил разработен за програмиране на битова електроника. В последствие Oak бил преименуван на език Java и започнал широко да се използва в разработването на приложения и сървърно програмно осигуряване.

През 2000-2001 години бил приет и стандартизиран нов език на програмиране C# (си-шарп), специално разработен за платформата .NET. В създаването на езика участвали програмисти от Microsoft Research (НИИ към корпорацията Microsoft) — Андерс Хейлсбърг, Скот Вилтамут, Питър Голде и други известни специалисти, в това число Ерик Майер. Версията на езика — С# 2.0 е представена пролетта 2005 година. Следва да отбележим, че един от езиците, положени в основата на С#, е все същият стар добър С++.

И така, от този кратък преглед на исторически факти става ясно, че С и С++ са два различни езика на програмиране, дори въпреки факта, че С++ е създаден на базата на С. Нашите занятия ще бъдат посветени на изучаване и на двата езика на програмиране. Естествено, получаването на знания ще става последователно и първо ще се запознаем с езика С, а после плавно ще преминем към С++.

И сега, след това малко пътуване в миналото, вече можем да се върнем в настоящето. За по-нататъшната ни работа безусловно ще ни е нужен съответният софтуер. Поради бързото развитие на пазара на IT-технологиите постоянно се създават все нови и нови версии на различни програми. В следващите раздели на урока ще разгледаме инсталирането на програмния продукт, който ще използваме в рамките на процеса на обучение.

# Инсталиране на Microsoft

Visual Studio 2013

Microsoft Visual Studio е пакет програмни продукти на Microsoft, които се използват за разработване на различни по масштаб решения: от лабораторни работи на студенти до проекти на корпоративно ниво. Visual Studio включва интегрирана среда за програмиране (IDE – Integrated development environment), многоброй-ни инструментални средства и помощни програми. С помощта на Visual Studio могат да се създават както конзолни приложения, така и приложения със сложен графичен интерфейс. Ние ще използваме продуктите на Visual Studio в рамките на процеса на обучение. Историята на Visual Studio започва през 1997 г, когато излиза Visual Studio 97. Вие ще изучавате програмиране с използване на Microsoft Visual Studio 2013. Това е най-актуалната версия към момента. Преди да разгледаме инсталирането на Visual Studio нека да видим въпроса с редакцията (edition) на Visual Studio. Редакцията Visual Studio — това е версия Visual Studio, която се различава с редица възможности. Visual Studio 2013 има следните редакции: Ultimate, Premium, Professional, Test Professional, Express Editions.Ultimate — това е най-пълната редакция на Visual Studio. Всяка редакция има своя ценова политика. Изключение е Express Editions. Това е пакет Visual Studio, създаден за

конкретни нужди. Например, за програмиране на мобилни решения или уеб-решения, или решения за работния плот и т.н. Express Editions може да се използва абсолютно безплатно, в това число и за търговски цели. Важно е да се знае, че различните редакции на Visual Studio са достъпни на различни езици. Например, може да ви се прииска да си инсталирате българска локализация. Не го правете! В работата си със сигурност ще се срещате с колеги и проекти от други страни, които вероятно ще използват англоезичен интерфейс. Именно затова е препоръчително винаги да предпочитате езика на оригиналната версия — английският.

Да разгледаме как се инсталира Visual Studio на примера на Microsoft Visual Studio 2013 Ultimate и Microsoft Visual Studio Express 2013 за Windows Desktop. Ще започнем от с Ultimate.

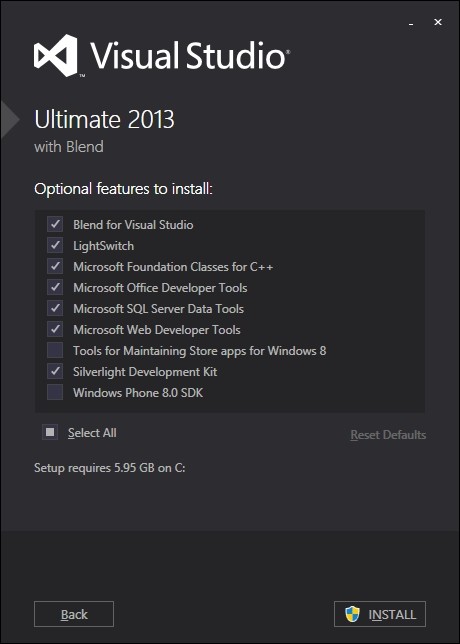
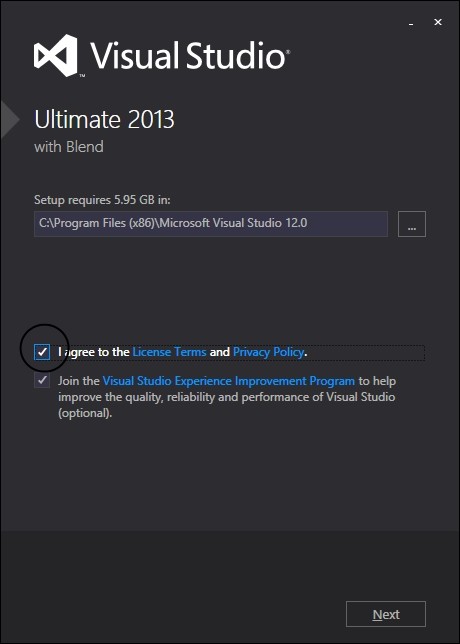


За да започне инста-лирането на Ultimate, трябва да поставим инсталационния DVD диск в устройството за четене (или да създадем ISO-образ на DVD диска например с помощта на програмата Daemon Tools Lite) и да отворим файла vs\_ultimate. exe, след което ще се отвори

следният прозорец (виж. снимката). В този прозорец може да изберете начин за инсталиране на Visual Studio, да решите дали искате да участвате в програмата за подобряване на Visual Studio

Ако искате да участвате в програмата за подобряване на качества, поставете отметка в опцията «Join the Visual Studio Experience Improvement Program…». За да започнете инсталирането на Visual Studio трябва да изберете «I agree to the License Terms and Privacy Policy».

Избирайки тази опция, вие потвърждавате, че сте запознати и съгласни с всички условия на лиценза. Ако всичко сте направили коректно, в долната част на прозореца ще се появи бутон «Next». Натиснете го. След това ще видите опция, която ви позволява да избирате допълнителни програмни продукти за инсталиране. На този етап може да оставите избраните по подразбиране продукти. По-късно ще имате възможност да добавите компоненти. За да продължите процеса натиснете бутона «INSTALL».

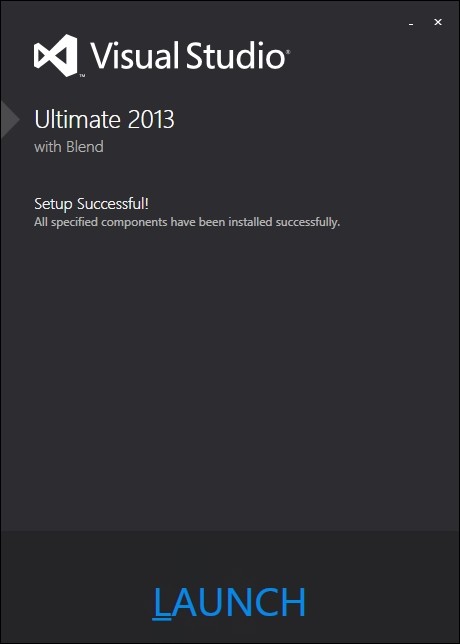


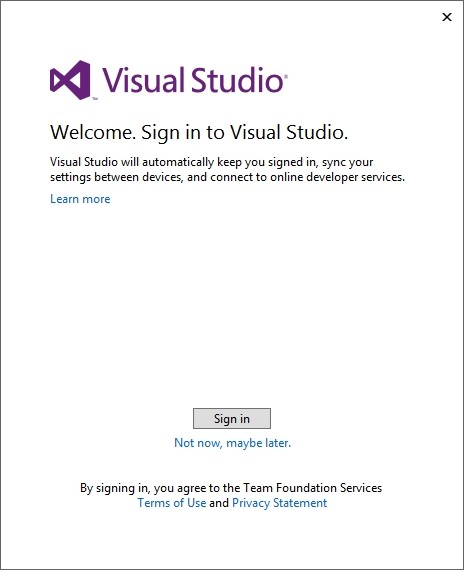
Вече трябва да е стартирал процесът на инсталиране. Това ще отнеме известно време, в течение на което вие ще можете да наблюдавате как се случват нещата. По-надолу даваме няколко характерни за това състояние прозорци.

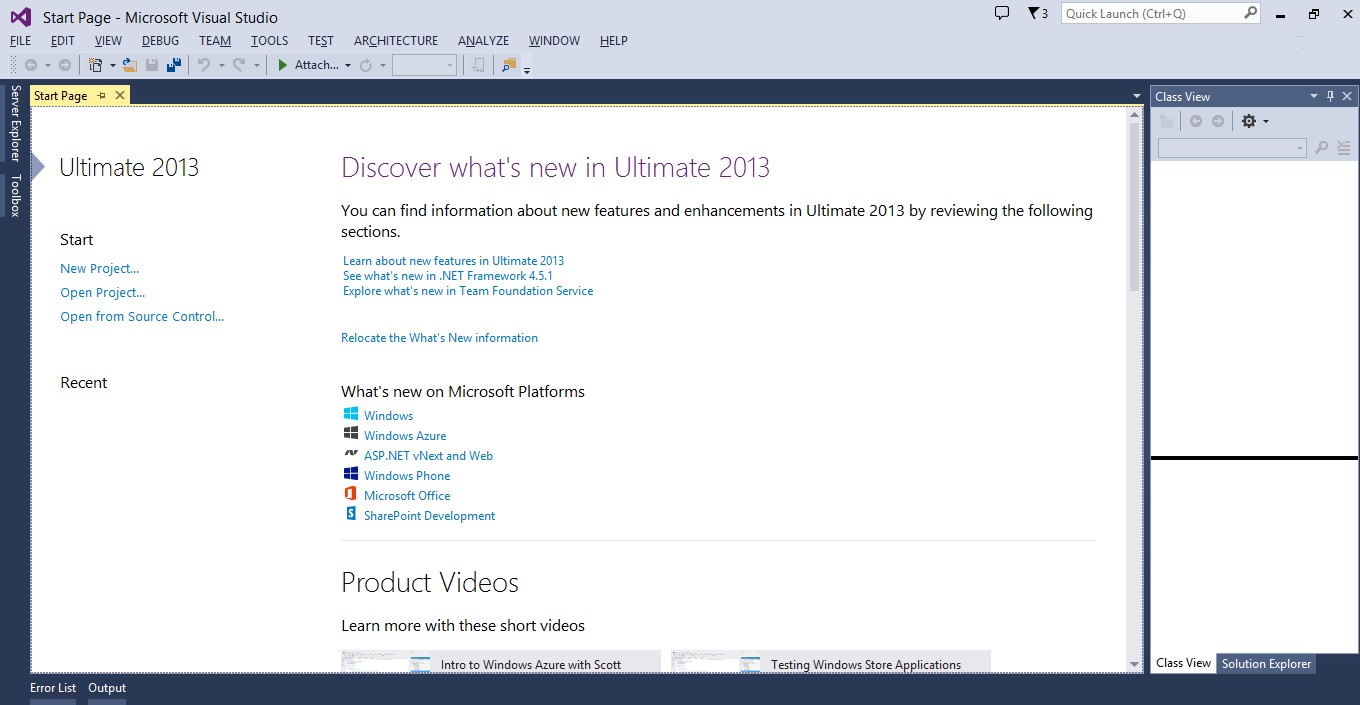
 

В края на инсталационния процес, ако всичко е минало успешно, ще видите прозорец с бутон «LAUNCH». С натискането му вие ще стартирате Visual Studio за първи път. Също така можете да използвате обичайния интерфейс за стартиране на Windows (менюто Старт, иконите и т.н.)

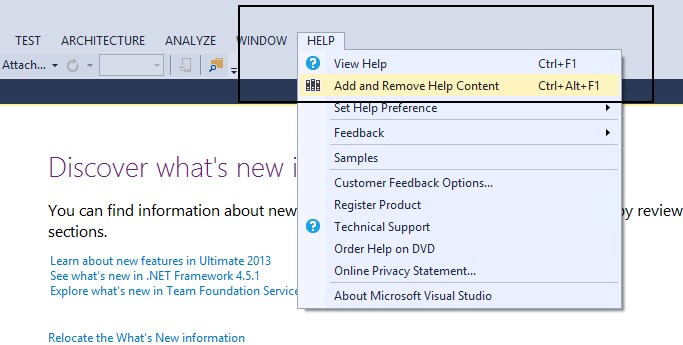
Нека стартираме Visual Studioза да се уверим, че предишните ни действия са били успешни. При първото стартиране, може да ви бъде поискано да въведете вашия Microsoft Live ID. При това имате и възможност да не го въвеждате. За целта необходимо да изберете точка «Not now, may be later».



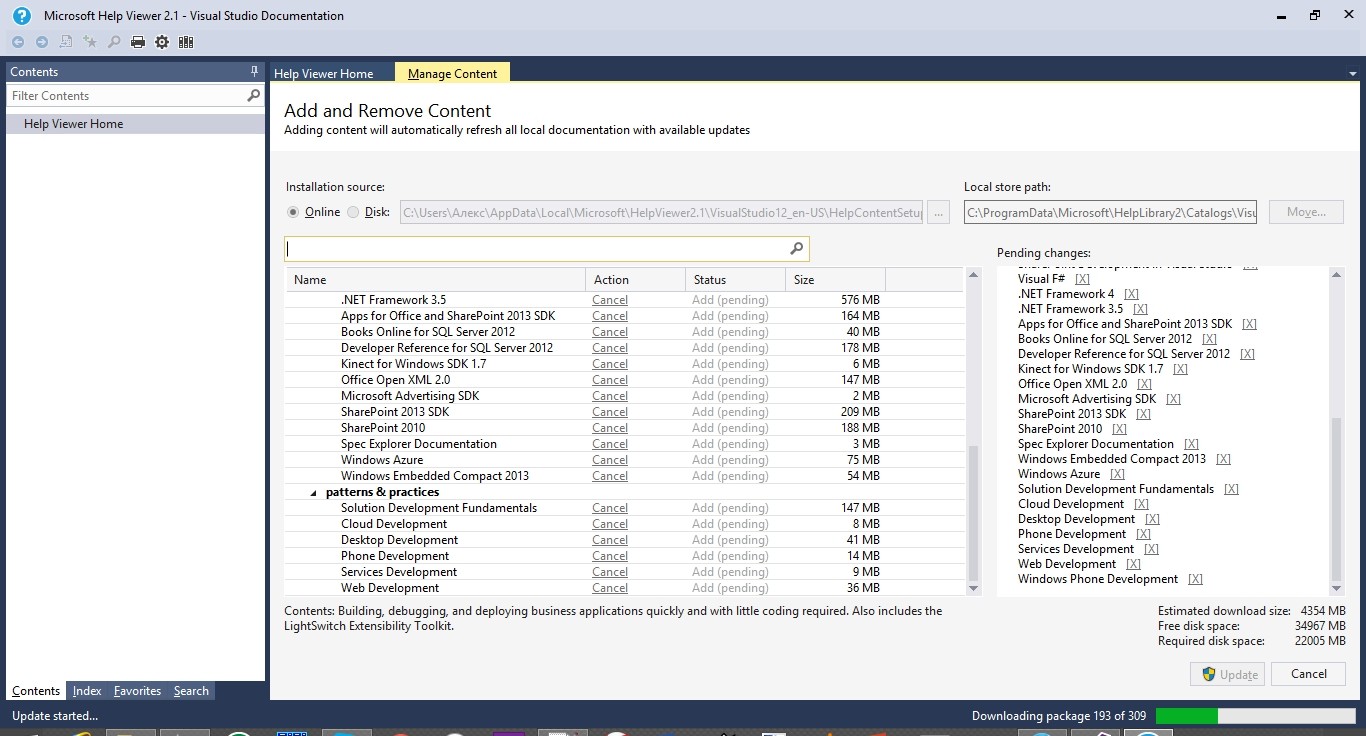
 



По време на обучението и разработването на приложения ще ви бъде много полезна справочната информация (MSDN) от Microsoft. За да си свалите локално копие на справката трябва да изберете опция от менюто H**elp->Add and Remove Help Content** или да натиснете **Ctrl+Alt+F1**.



Ще видите прозорец с избор на справочен материал. Препоръчваме навсякъде да кликнете «Add». Справките никога няма да са ви излишни :) След като направите избор на материали кликвате върху бутона «Update» за да ги запишете на нужното място.



Сега да разгледаме процеса на инсталиране на Microsoft Visual Studio Express 2013 за Windows Desktop. Той много прилича на инсталирането на Ultimate.

За да стартирате инсталирането на Express, поставете DVD диск в оптичното устройство (или създайте ISO-образ на DVD диска с помощта на, примерно, програмата Daemon Tools Lite) и стартирайте файла **wdexpress\_full.exe**, след което ще излезе следният прозорец:

Ще можете да изберете пътя за инсталиране на Visual Studio, да решите дали искате да участвате в програмата за подобряване на Visual Studio. Ако желаете да участвате в програмата за подобряване на качеството на процедурите по инсталиране на даден продукт, отбележете опцията «Join the Visual Studio Experience Improvement Program». За да започнете процеса на инсталиране на Visual Studio ще трябва да изберете «I agree to the License Terms and Privacy Policy».

Избирайки опцията, вие потвърждавате, че сте запознати и съгласни с условията на лиценза. Ако всичко сте направили коректно, в долната част на прозореца ще се появи бутон «INSTALL». Натиснете го и процесът на инсталиране на Visual Studio ще започне. Това ще отнеме известно време, през което вие ще можете да наблюдавате как се случват нещата. По-надолу даваме няколко характерни за това състояние прозорци.



В края на инсталирането, ако всичко е минало успешно ще видите прозорец с бутон «LAUNCH». Натискайки го, ще стартирате Visual Studio за първи път. Също така можете да използвате обичайния интерфейс за стартиране на Windows (менюто Старт, икони и т.н.). Както и при инсталиране на Ultimate редакцията ще ви поискат да въведете своя LIVE ID (може да не се въвежда), също да не забравите да инсталирате и справочната система.

Ако сте преминали всички етапи на инсталирането успешно, вече сте готови да напишете своята първа програма.

Това е. Сега вече можете да преминете към следващите раздели на урока, където ще се учим да програмираме.

# Първи проект

Един човек на име Сергей, описвайки своите впечатления за Прага, ми беше написал:

«Ако ви се падне сервитьор, който не говори руски, хващайте менюто и сочете с пръст какво искате. Понякога думи, които изглеждат познато, могат да означават различни неща. Например, думата за плодове на чешки звучи почти като зеленчуци[[1]](#footnote-1). Познайте, какво ще ви донесат, ако си поръчате зеленчукова салата?:)»

Неизвестният за Вас Сергей иска да предпази своите приятели от възможна грешка, но едва ли би могъл да помогне, ако вместо със сервитьор ни се налага да общуваме с компютър. Последният, уви, нее е наясно кой точно ред от менюто сочите с пръст. Компютърът педантично следва само конкретни указания. И тези указания трябва да се дават с помощта на специални команди. Командите на свой ред съставляват цели самостоятелни езици. Езиците, понятни за компютъра се наричат езици на програмиране. Извод: за да намерите общ език със сервитьора чех, трябва да сте находчив. За да намерите общ език с компютъра,трябва да знаете езика на програмиране.

От първия раздел вече Ви е известно, че C — това е език на програмиране. Език, който ни дава възможност вразумително да обясним на компютъра какво точно искаме от него. Макар, ако трябва да сме откровени до край, компютърът разбира само един език — езика на машинните кодове. Например програмата, която извежда на екрана фразата «Hello, world!», изглежда приблизително така на «родния» за компютъра език:

\_YН¦ў+.\_5lЧ№+f-Н¦ў+ +f+ ¤д-ы+fО\_fО¦¤+эm№¤+¤+¦Ф№+¤-;-wGГ=xФ№+MZР + @ А -¦ ¦ -!+ L-!T u .\_5lЧ№+f-&ЛZ¶QW&бфЬ№+&Л &ЛJf&ОB\_Ф№

+f;sЛlФ№+Л&lt; \_t=ИФ№+\_G¶u\_O¶WfМ+-чfЛ№3+М-Ас°

\_>] Л¦Л+ЛMЛm \_сбшЬ№+Л 9ШФ№+t§Л§pФ№+ш№) fd¦ +RPбшЬ

№+Л 9ШФ№+tЛ§pФ№+шС) + +Э

tАс°-щЗ-Л\_Ф№+f;s §lФ№+Л +u"бАФ№+¤+MАс°ЎD Аt Э+- 3+ ыўы•ы°ыЎыЇЛD$П$.\_4Ч№++ .\_ dЛ4 Ў- ubSЛ dг4 + є --fУЙf-+

-+ЙA+@У№+Л-[Xf +tRPRVh•ў+ш\_S Z+єfО-fЛцf-d¤+§ RfRfh ш4 Л+Zы¦SшC¤д-ы¦¤+ьМ-¤++¤\_e fМ+fО\_fО¦f¤бЛЁdЎ t3+ОшАс°Л§АФ№+- dб4 ЙdЙ§4 f\_f \_t dЛ4 Ў- uSSЛ dг4 + є --fУЙf-+-+ЙA+@У№+Л-[

+єЙ]№fМ+fО-fЛц+ыf-d¤+§ RfRfh шз Л+Zы+Sш¦

$ PE L ¦7 р ¤!

` Р oK Р ў+ 0

Ще кажете, че на такъв код не е възможно да се пище и ще сте абсолютно прави! Но ния няма да пишем така. За това е и нужен език програмиране, който да улеснява съставянето на програми. Езиците на програмиране се делят на две основни групи ИНТЕРПРЕТИРАНИ и КОМПИЛИРАНИ – в зависимост от това, каква специализирана програма превежда командите от езика на програмиране на машинен език — КОМПИЛАТОР или ИНТЕРПРЕТАТОР. Нека да изясним каква е разликата между тях. Да си представим, че имаме файл, в който се съдържа набор команди

## Ситуация първа. Командите са написани на интерпретиран език.

Всеки път при стартиране на програмата интерпретаторът прави проверка на кода ред по ред. Ако няма грешки в синтаксиса, командите ще бъдат преобразувани в машинен код (набор инструкции за процесора). Програмата ще стартира за изпълнение. Ако има грешка, интерпретаторът ще спре и ще Ви бъде предложено да я поправите и да стартирате програмата отново. Но, дори ако грешката вече поправена и програ-

Мата е окончателно дописана, при всяко нейно стартиране интерпретатор ще прави проверка на кода. Изводът е, че машинната версия на кода никъде не се съхранява. Минусите на този подход са в това, че скоростта на стартиране на програмата намалява, но не е възможно да се изключи проверката на кода.

## Ситуация втора. Командите са написани на компилиран език.

Компилаторът действа почти като интерпретатора, т.е. проверява кода ред по ред. Но ако среща грешка, той не спира, а продължава да изследва кода до края като открива всички следващи грешки и изкарва съобщения за тях. Освен това компилаторът формира специален обектен файл с разширение .OBJ. В този файл се съхранява текстът на програмата, преведен на машинен език.

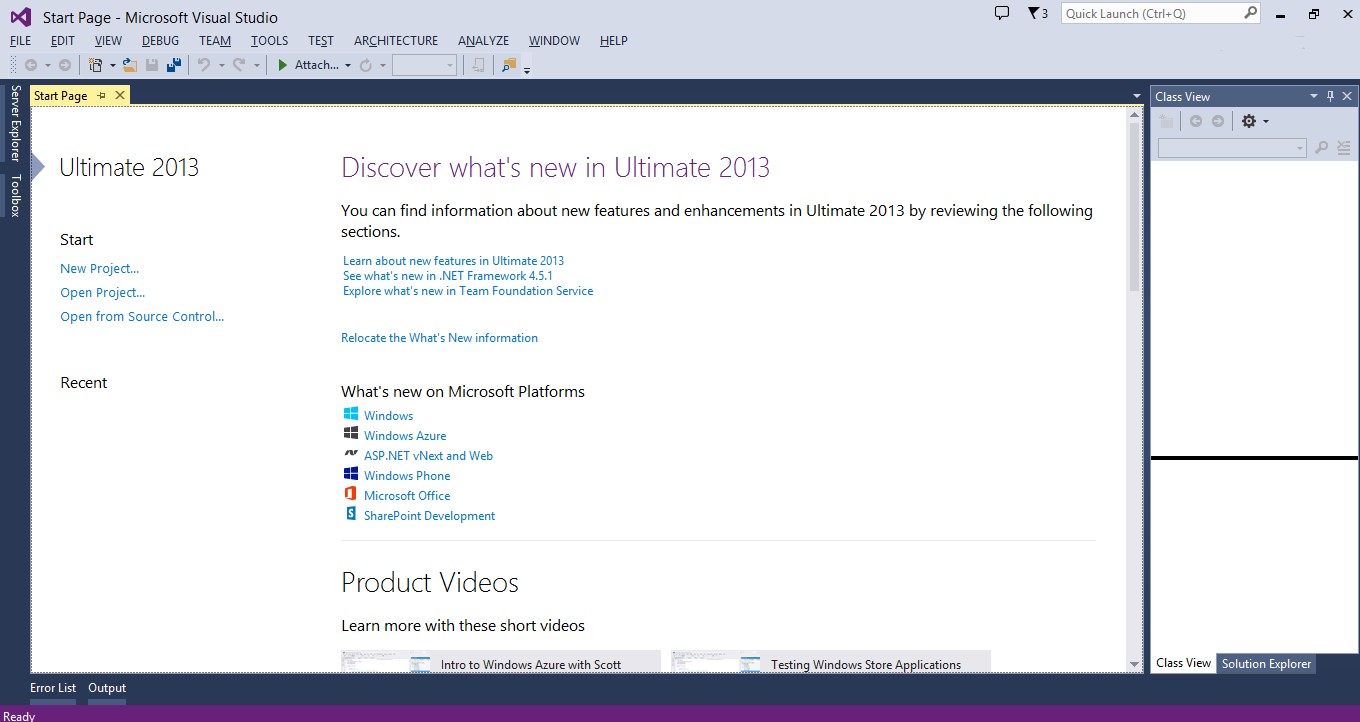
Обаче, компютърът не работи директно с този файл. Съществува понятието компониране или свързване. Свързващият редактор е още една специална програма, която сглобява машинния код (от файла с разширение .OBJ) и различни спомагателни данни в единен изпълнителен файл с разширение .EXE. Такъв файл може да бъде стартиран за изпълнение като отделна, самостоятелна програма и компилаторът вече не участва в стартирането й.

Езикът С, който започваме да изучаваме, е компилиран език. В нашия случай при работа с Microsoft Visual Studio обръщението към компилатора става автоматично и позволява преводът на командите на езика С в машинен код, да става , както се казва , «с лека ръка».

## Първи опит

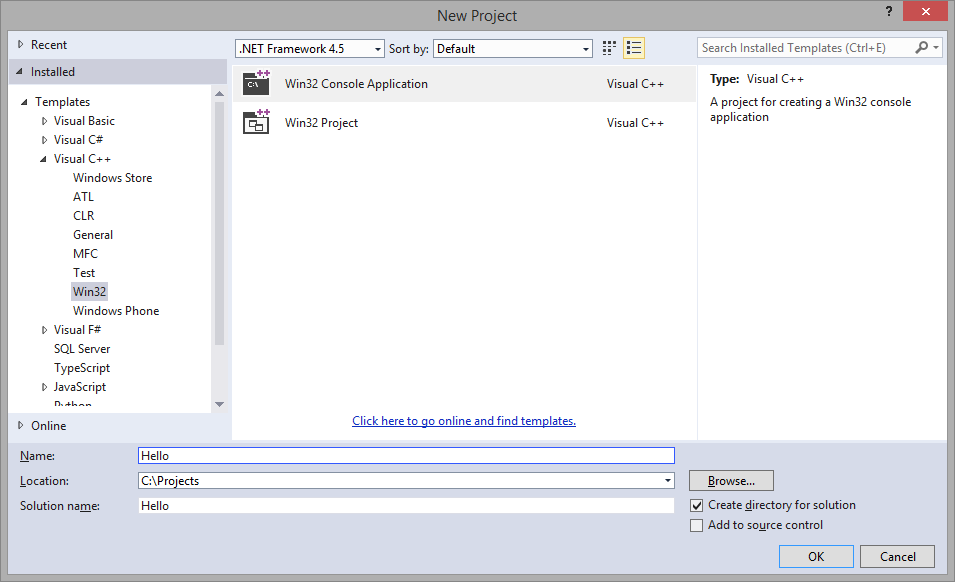
Един от основателите на езика С, Браян Керниган, казал:«Единственият начин за изучаване на нов език на програмиране е да пиеш програми». И ние с Вас ще започнем да правим точно това. Традиция е в света на програмирането, първата програма на всеки нов език да е програмата «Hello world!».

За да напишете своята първа програма най-напред трябва да стартирате иконата на програмата Microsoft Visual Studio 2013 от опцията на меню «Старт» -> «All Rrograms» -> «Microsoft Visual Studio 2013» или по друг привичен за вас начин. След стартирането ще видим изображение, подобно на представеното по-долу:

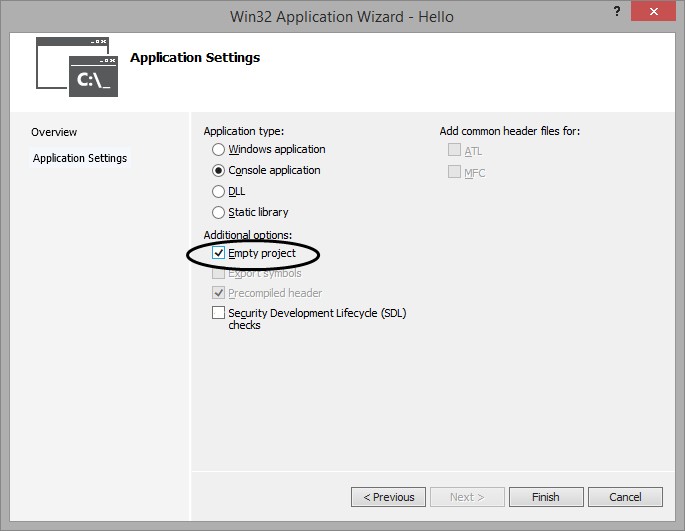


Сега да опитаме да създадем проект, който в крайна сметка ще е нашата програма. По-подробно за това, що е то проект, ще говорим по-късно, когато ще пишем големи програми. Засега да си представим проекта като обединение на няколко файла. И сега – стъпка по стъпка:

* След като стартирате програмата, избирате опция от менюто File -> New -> Project. Ще се отвори диалогов прозорец.
* В отворилия се диалогов прозорец New Project (нов проект) в списъка Installed Templates (шаблони) изцерете вида проект Visual C++ -> Win32 -> Win32 Console Application
* В полето Location посочваме, на кой диск и в коя папка ще се намира проектът ни. За това напишете в това поле C:\Projects (или удобно за Вас име на папка, в която ще се съхраняват всичките Ви домашни проект)
* Да дадем име на нашия ПРОЕКТ — за целта въвеждаме в полето Name името на проекта Hello.
* Сега вече можем да натиснем бутона OK.

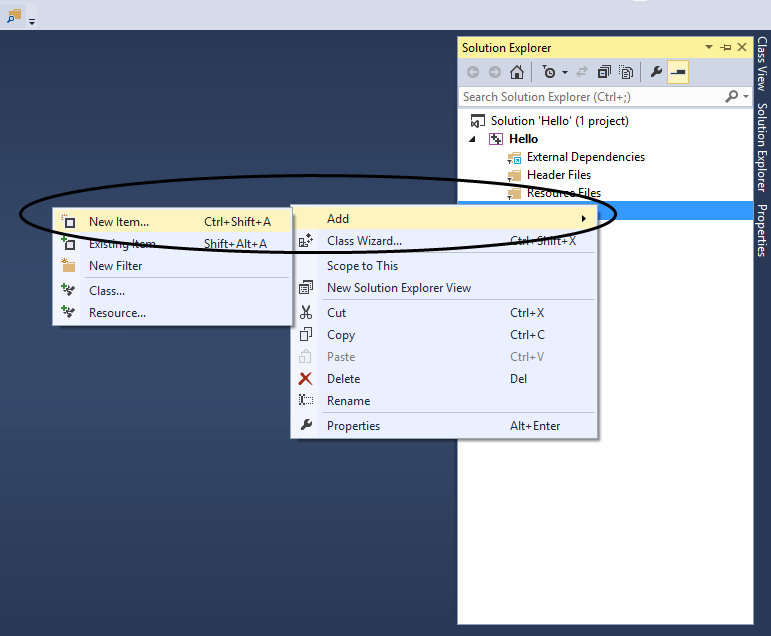


Пред Вас се отваря прозорец за настройки на свойствата на проекта — изберете опция Application Settings. Отбележете полето **Empty Project** — това значи, че създаваме празен проект. Също премахнете отбелязването от **Security Development Lifecycle (SDL) checks**. Сега натиснете бутона Finish.

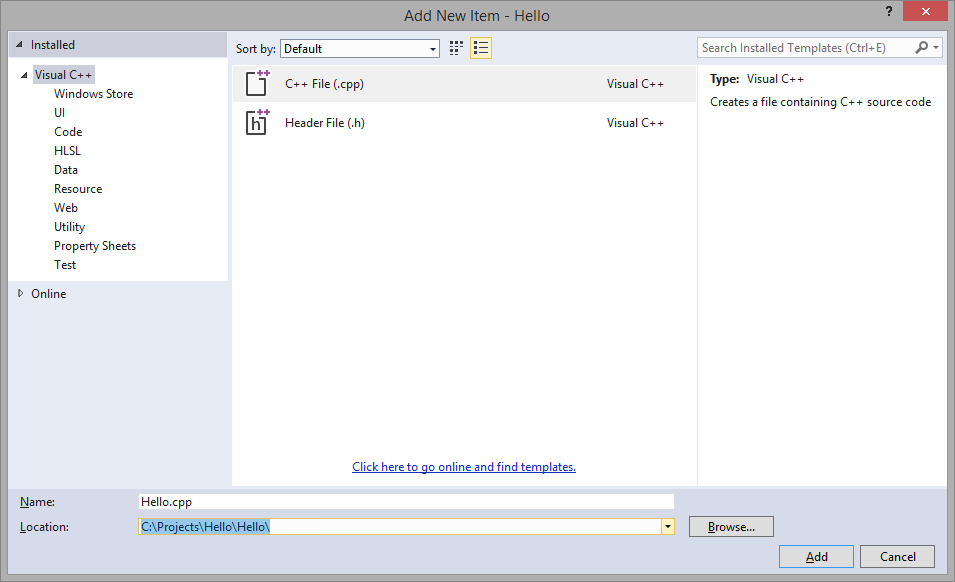


И така, подготвили сме място за разполагане на нашата програма. В така създадения проект добавяме чист файл. В него ще набираме текста на нашата програма. За целта е необходимо да изпълним следните действия:

* Отдясно виждаме прозореца Solution Explorer. За да се появи този прозорец може също да се използва комбинацията от клавиши Ctrl+Alt+L. В този прозорец е необходимо да щракнете с десния бутон върху папката с название Source Files.
* От падналото меню избираме Add->Add New Item...



* Отваря се прозорец за избор на файлове. Отново имате огромен избор. Препоръчително е да изберете знака C++File (.cpp) (файл, съдържащ програма, написана на езика С/C++).
* В текстовото поле Name (име на файла) напишете името на файла Hello. Натиснете бутона Add.



След като натиснете бутона Add ще се появи текстова област, където ще можете да набирате своята първа програма.

## Пример на първа програма на езика C.

Преди да започнете да пишете програмата, трябва да въведем за удобство понятието за коментар. Коментари наричаме бележки към програмата, предназначени изключително за програмиста. Компилаторът ги игнорира. Например с помощта на коментар може да се обозначи за какво да се използва този или онзи ред от програмата. В описания по-долу пример е посочено как правилно да се работи с коментарите.

А сега в появилия се текстови прозорец ще наберем следния код:

// Това са коментари към програмата

// Те се маркират със зелен цвят

// Започват коментарите с две наклонени черти //

/\* ако е необходимо да се създаде многореден коментар, използва се конструкция

/\* коментар \*/

/\* Този ред включва в програмата библиотека с наименование iostream. Библиотеката е файл, в който се съдържат описания на различни функции, реализирани от други програмисти.

Тази програма е получила възможност да използва функциите, намиращи се в библиотеката iostream \*/

#include <iostream>

/\* В езике С++ съществува понятие именно пространство.

Това пространство определя някаква област за действия на оператор или функция.

За да се използва оператор, намиращ се в определено пространство,

необходимо е това пространство да се включи към твоята програма.

По-долу се включва пространство наречено std \*/ using namespace std;

void main() // Начало на програмата, оттук програмата ще започне да се изпълнява

// Целият текст на програмата се разполага между фигурни скобки

{ //Това е фигурна скобка

// Следващият ред извежда на екрана приветствието Hello, World!

// Това действие се осъществява с помощта на cout<< Именно за неговата

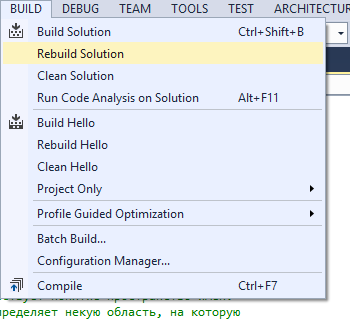
// работа са включени библиотеката и пространството на имена, в което то се разполага cout<<"Hello, World!\n";

// В края на командата се поставя точка със запетая. С този знак ТРЯБВА ДА завършва

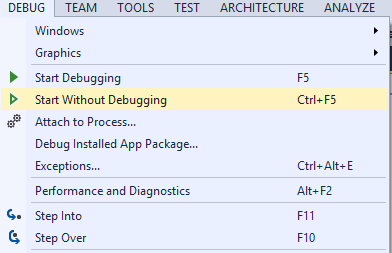
// всяка команда в езика С.

}

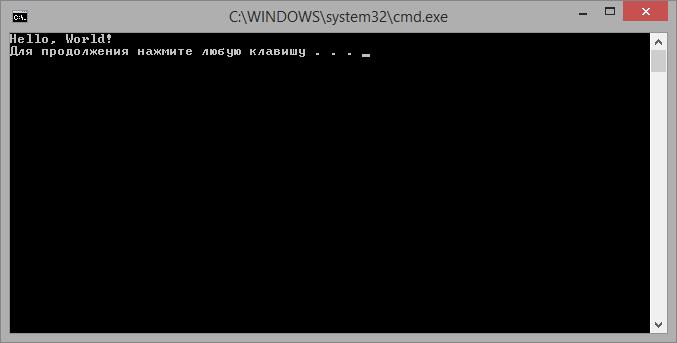
Както вече знаем, компютърът разбира само езика на машинните кодове. И преди програмата да започне да се изпълнява от компютъра, тя трябва да се преведе на езика на машинни кодове. Спомняте си, че това няма да го правим ние, а КОМПИЛАТОРЪТ. Проектите, написани на C, включват множество файлове. Ще компилираме наведнъж всички файлове, включени в проекта. За целта в реда на менюто избираме опцията Build, а после Rebuild Solution



Надяваме се, че текстът е набран без грешки. Нашата програма е успешно преведена на езика на машинните кодов и можем да я стартираме за изпълнение. Това става много лесно. В менюто Debug изберете Start Without Debugging.



Програмата ще тръгне и ще видите ето такъв прозорец:

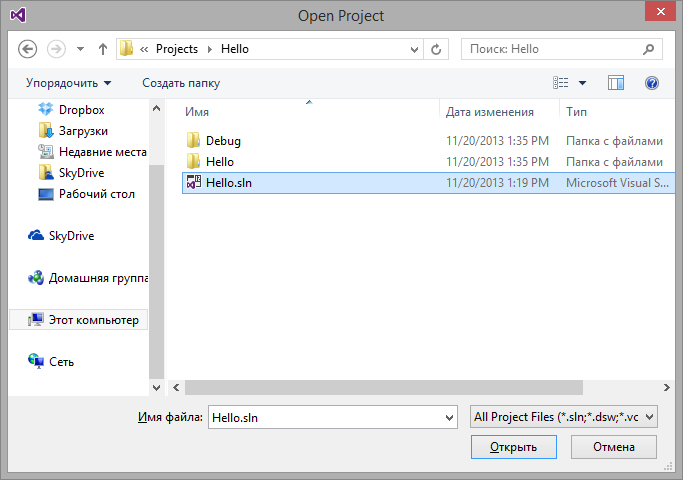


Честито! Това беше лесно. Нали така? За да приключите работата на програмата натиснете произволен клавиш на клавиатурата (Press any key to continue).

## Намиране на запазения проект

За да възстановите запазен по-рано проект на диска стартирайте Visual Studio (ако още не сте го направили). Изберете в менюто File опция Open ->Project/ Solution и посочете името на вашия проект.





След като намерите проекта, можете да продължите работата си по него. Проектите, които създаваме в рамките на урока, ще можете да намирате в подпапката на урока с наименование Sources. В частност този проект се намира на пътеката Папка с урока/Sources/Hello.

# Извеждане на данни

Вече знаем, че с помощта на команда cout<< можем да извеждаме на екрана различни текстови редове. Обаче за да разбере компилаторът една такава команда, необходимо е да помним за следващите три основни момента:

1. В заглавието на програмата трябва да присъства редът #include <iostream>
2. Преди да се използва командата необходимо да бъде включено именното пространство, към което принадлежи командата cout.

**using namespace std;**

1. Реда, който искаме да изкараме на екрана, използвайки cout<<, задължително се поставя в кавички. Например:

**cout<<”здесь пишем то, что хотим”;**

Командата cout<< не само извежда на екрана редовете, но и позволява те да бъдат оформяни. За оформяне на извеждания на екрана ред използват специални управляващи символи - комбинация от символа\и символа, определящ действието, което трябва да се осъществи с реда. Тези управляващи символи се наричат Escape-последователности. По-долу са дадени някои от тях:

**\b**

**\n**

**\t**

**\\**

**\"**

**\'**

**Премахване на последния изведен символ**

**Премини към на начало на новия ред**

**Премини към следваща позиция на табулатора**

**Изведи обратна черта \ Изведи двойна кавичка “**

**Изведи единична кавичка '**

Съществуването на последните три Escape-последователност първоначално винаги предизвиква леко недоумение. Защо да използваме управляващи символи, ако можем просто да напишем: " или \ или ' ? Отговорът е на повърхността: и трите символа са оператори и ако «просто ги напишете», компилаторът ще ги възприема именно като оператори. Например когато една дума се използва в преносен смисъл, тя се поставя в кавички. Да предположим, че искаме да изведем следния текст на екрана:

**The Man in red was "old friend" of John...**

Ако не използваме Escape-последователности, очевидно е, че командата ще изглежда така:

**cout<<”The Man in red was "old friend" of John...”;**

И това неминуемо ще доведе до грешка. Компилаторът ще възприеме само част от реда, а именно cout<<''The Man in red was". Двойната кавичка след was той ще сметне за затваряща, а всичко останало ще отчете като грешка в синтаксиса на езика. Такава програма, естествено, няма да тръгне за изпълнение. Правилният вариант е такъв:

**cout<<”The Man in red was \"old friend\" of John...”;**

А сега нека видим къде именно в cout<< могат да се поставят Escape-последователностите. Най-важното е да се знае, че Escape-последователността винаги трябва да е затворена в кавички, тъй като това е текст, а после възможностите са практически не ограничени. Например, така:

cout<<'' My name is''<<'' - Ira\n ''; cout<<''I'm from Odessa\n '';

cout<<''My eyes are blue"<<"\n ''<<"That`s all!!!";

В резултат на работа на тази команда ще видим на екрана:

**My name is - Ira I'm from Odessa My eyes are blue That`s all!!!**

## Практически пример за използване на cout<<

Да напишем програма, която извежда на екрана кратка справка за това как ние с вас изучаваме Escape-последователностите. Ето какво искаме да видим на екрана:

**\b**

**\n**

**\t**

**\\**

**\"**

**\'**

**Backspace New line Horizontal tab Backslash \**

**Double quotation mark " Single quotation mark '**

Стартираме **Visual Studio 2013**. Създаваме нов проект с наименование **EscapeSequences**. Набираме код, който ще видите по-долу.

// Заглавие

#include <iostream>

// определяне на именното пространство, в което има cout<< using namespace std;

// Главна функция void main()

{

/\* Следващата команда след 4 табулации извежда текста Escape Sequences

и прехвърля извеждането на следващия ред \*/

cout<<"\t\t\t\tEscape Sequences\n";

// Извежда празен ред

cout<<"\n";

/\* След 2 табулации извежда текст \b, и още след 1 табулация Backspace

После \n прехвърля извеждането на следващия ред \*/

cout<<"\t\t\\b"<<"\tBackspace\n";

// Извежда празен ред

cout<<"\n";

/\* След 2 табулации извежда текст \n, и още след 1 табулация New line

После \n прехвърля извеждането на следващия ред \*/

cout<<"\t\t\\n"<<"\tNew line\n";

// Извежда празен ред

cout<<"\n";

/\* След 2 табулации извежда текст \t,

и още след 1 табулация Horizontal tab

После \n прехвърля извеждането на следващия ред \*/

cout<<"\t\t\\t"<<"\tHorizontal tab\n";

// Извежда празен ред

cout<<"\n";

/\* След 2 табулации извежда текст \\, и още след 1 табулация Backslash \

После \n прехвърля извеждането на следващия ред \*/

cout<<"\t\t\\\\"<<"\tBackslash \\\n";

// Извежда празен ред

cout<<"\n";

/\* След 2 табулации извежда текст \t,

и още след 1 табулация Double quotation mark "

После \n прехвърля извеждането на следващия ред \*/

cout<<"\t\t\""<<"\tDouble quotation mark \"\n";

// Извежда празен ред

cout<<"\n";

/\* След 2 табулации извежда текст \',

и още след 1 табулация Single quotation mark '

После \n прехвърля извеждането на следващия ред \*/

cout<<"\t\t\'"<<"\tSingle quotation mark \'\n";

// Извежда празен ред

cout<<"\n";

}

Сега компилираме програмата **(Build -> Rebuild Solution)**. Ако има много грешки, припомнете си следните правила:

* Ако в програмата ще се извеждат съобщения на екрана, то в началото на програмата се записва реда
* #include <iostream> и се включва именно пространство, към което се отнася командата cout (using namespace std;) Всяка програма трябва да съдържа функция с името main (). Работата на програмата започва от изпълнението на тази функция.
* Командите за функцията main() се включват между две фигурни скобки { }
* Всички команди задължително трябва да завърпват със символа „точка със запетая“.
* И запустим ее **(Debug -> Start Without Debugging)**.

## P. S.

Сигурно сте обърнали внимание, че използваме само латински символи при извеждане на екрана на данни. Това е така, защото програмата пишем в ОС Windows, а изпълнението й се осъществява в MS DOS. Работата е там, че всеки символ има свой числови код независимо от операционната система. И системата го идентифицира именно по този код. Символните кодове на кирилицата в MS DOS и Windows не съвпадат, поради което програма с използване на кирилица няма да работи коректно. Например, ние пишем в Windows:

**cout<<"Утро";**

А на екран излиза:

**µЄЁю**

Обяснението се състои в това, че в Windows например буквата о е 238, а в DOS на този код отговаря буквата ю. Докато кодовете на латиницата съвпада в двете ОС. В последствие ще се научим как да излизаме от такава ситуация.

В стандарта на езика C++ 11 програмистите получиха нова възможност за извеждане на специални символи на екрана. За това е необходимо да се използват «raw» редове. Обявяването на реда за «raw» дава указание на компилатора да го проверява посимволно. За да бъде един ред обявен за «raw» необходимо е да се използва следния формат.

**R"(текст\_строки)"**

R показва, че това е «raw» ред. Съдържанието на реда задължително се затваря в скоби. Да дадем няколко примера:

cout<<R"(hello\nworld)"; // на экране hello\nworld cout<<R"("Test 'string'\t")"; // на экране "Test 'string'\t" cout<<R"((Such brackets))"; // на экране (Such brackets)

# 5. Видове данни

След като вече сте прочели предните раздели на урока, вече нищо не ви струва да напишете програма, която да извежда нещо екрана.

// Заглавие

#include <iostream>

// определяне на именното пространство, в което има cout<< using namespace std;

// Главна функция void main()

{

// извеждане на израза "Честито добро начало!!! :)"

// изразът се извежда след три табулации,

// после добавяме два празни реда cout<<"\t\t\tCongratulation with good beginning!!! :)\n\n";

}

И това е всичко, което умеете (засега).

Човек никога не трябва да се спира на постигнатото. Собствено би трябвало да се заинтересувате не само от това как се извеждат данни на екрана, но и как да се оперира с тези данни. Например да се правят някакви изчисления. Безръки жонгльори няма, и, докато част от топките са във въздуха, останалите са в ръцете на артиста. За да се съхранява нещо (в частност данни) необходимо е да имаме хранилище. За нашата програма такова хранилище ще бъде оперативната памет. Преди да разполагаме нещо някъде, необходимо е да подберем подходяща опаковка. Да кажем, че едва ли бихте сипвали мляко в кибритена кутия. В програмирането, преди да се разполагаме информация в оперативната памет, трябва

Задължително да сме определили характера на тази информация.

И така, **Видове данни** е понятие, определящо максималния размер (в байтове) и типа информация, която ще се ползва от програмата.

Програмирането отчасти отразява обекти от външния свят, като ги опростява изрядно. В началото на обучението ние ще се срещнем с неща, с които Вие сте се сблъсквали много пъти. Хайде условно да разделим всички типове данни на следните групи.

* 1. Числови. .
  2. Символни.
  3. Логически.

По-нататък ние ще разгледаме няколко ключови думи , използвани в езика С за обозначаване на типовете данни.

## Числови типове.

Числата, както е известно биват цели и веществени. Веществени числа ще наричаме числа с **плаваща** точка. Ще отбележим че, запетаята която разделя цялата част от дробната се заменя с точка. Например 7,8 в езика С се пише 7.8

Променливи в които ние ще пазим значението веществени числа ще се обявяват с типове float или double. Каква е разликата между тези типове? Тип float описва числа с плаваща точка и единична точност, а double- с двойна точност. За пояснение ще кажем, че в математиката точността се определя от количеството цифри които представят числото. Двойна точност се нарича метод за представяне на число което е с двойно по голямо количество цифри от обикновеното. Ето характеристика за типовете числа с плаваща точка:

|  |  |  |
| --- | --- | --- |
| Пояснение | Тип | Размер в байт |
| описва веществени числа с единична точност | float | 4 |
| описва веществени числа с двойна точност | double | 8 |

Освен вечестени в С са предвидени три типа обявяващи целочисленни данни. В таблицата са приведени основните характеристики на тези типове:

|  |  |  |  |
| --- | --- | --- | --- |
| Пояснение | Тип | Размер в байтах | Диапазон значений |
| описва цели числа | int | 4 | от -2147483648 до  2147483647 |
| описва кратки цели числа | short | 2 | от -32768 до 32767 |
| описва дълги цели числа | long | 4 | от -2147483648 до  2147483647 |
| описва дълги цели числа | long long | 8 | от –9,223,372,036,  854,775,808 до 9,223,372,  036,854,775,807 |

## Символен тип.

Тип предназначен за запазване само на един символ. Веднага ще предупредим – тип за запазване на линии в С не съществува.

|  |  |  |
| --- | --- | --- |
| Пояснение | Тип | Размер в байтах |
| описва символи | char | 1 |

## Логически тип.

Тип предназначен за запазване на логически данни. Подробно ще се запознаем с тях по- късно. Логическите данни могат да приемат една от двете значения: вярно (true) или грешно (false).

|  |  |  |  |
| --- | --- | --- | --- |
| Пояснение | Тип | Размер в байтах | Значения |
| описва логически значения | bool | 1 | true false |

**Забележка:** Ако е необходимо да се изключи от диапазона типове данни с отрицателно значение, пред названието на типа трябва да се сложи ключовата дума unsigned. Например unsigned int. Такъв тип ще включва в себе си само положителни стойности от 0 до 4294967294.

И така ние изяснихме какви биват типовете данни и кои ключови думи от езика С се използват за да бъдат обозначавани. За заключение трябва да отбележим, че езика С е регистрозависим, тоест думи написани с големи и малки букви не са с еднакво значение. Обърнете внимание на това че всичките гореописани типове данни са записани малки букви. Следете за това , след като int- това е тип данни, а INT- грешка.

В следващата тема ние ще разгледаме прилагането на типове данни практически.

# 6. Променливи и

константи

Ние с Вас вече се запознахме с типовете данни и знаем как се класифицира информация за запазване. Остана да изясним как да запишем данните в оперативната памет и как после да се обръщаме към тях за да ги използваме или променим.

И така, променящите се данни ще се договорим да ги наричаме ПРОМЕНЛИВИ, а постоянните данни КОНСТАНТНИ. Променливи- област на оперативната памет която има собствено име и е предназначена за запазване на данни които могат да бъдат променени.

Константа – област от оперативната памет която има собствено име и е предназначена за запазване на постоянни данни.

Ето пример за константа: На всички им е известно количеството дни в седмицата и количеството месеци в годината. Те не се променят при никакви обстоятелства- затова тези значения са константи.

А нашата възраст- е променлива величина. Днес някой е на 26 години, а след година ще бъде на 27.

От определенията става ясно че за търсене на данни в паметта им се дават имена (по аналог с този, както в багажното отделение на вагоните на вещите им се слагат стикери). В средата на програмирането тях ги наричат идентификатори. Един от първите проблеми който решават родителите с новородено дете това е избора на име за него. Наслагва ли името отпечатък на характера на човека, на неговата съдба е сложен и спорен въпрос. Може да се чуят съвършено противоположни мнения по този въпрос. Но факта, че името- идентификатора дадено на нова променлива (константа), трябва да бъде интуитивно разбираемо и да обяснява назначението на променливата, няма да го оспори нито един по-опитен програмист.

Имена на данни се дават със съблюдаването на строго определени правила. Тези правила не трябва да се нарушават!

## Правила за съставяне на имена.

В името е допустимо използването само на следните символи:

1. Големи и малки букви от латинската азбука. При това не забравяйте за регистрозависимостта на езика. Например, Age и age — това две различни имена.
2. Цифри. Обаче, цифра не може да бъде използвана в качеството на първи символ. Тоест, Name1 е допустимо, 1Name — не.
3. Символ за подчертаване «\_». Работата е такава че трябва да помните, че space също е символ и този символ е недопустим в името на променливата. Него ще го заменим със знак за подчертаване, който ще подобри изразителността на името. Например, сравнете: ageofman и Age\_Of\_Man.

При определяне на име за променливата запомнете следното:

1. Не трябва променливите да бъдат наименувани с думи които са ключови за езика на програмирането. Ключова дума— дума, резервирана в синтаксиса на езика за програмиране (int, float,double и др.). В Visual Studio ключовите думи са оцветени в синьо, това най-малкото ще доведе до объркване.
2. Не е желателно съществуването на два идентификатора с еднакви имена.
3. Не трябва да се използват никакви други символи, освен допустимите.

## Обявяване и използване на променливи и константи.

Сега вече имаме цялата информация за създаването (обявяването) на променлива. Остава само да се изясни какъв е общия синтаксис:

1. **тип\_на\_данни име\_на\_променливата**; — в този случай в оперативната памет ще бъде отделена клетка с размер съответстващ на зададения тип. Тази клетка ще присвои избраното от Вас име. Какво ли ще се съхранява там? В току-що създадената променлива ще бъде записано случайно число, определяемо от операционната система. Това число ще бъде в паметта докато не запълните променливата с друго значение с помощта на специален оператор за присвояване =
2. **тип\_на\_данни име\_на\_променливата = значение**; — съществува и такава възможност – да се запълни променливата с значение веднага при създаването. Такъв процес ние ще наричаме инициализация.
3. **const тип\_на\_данни име\_на\_променливата =значение**; — а това обявява константите. Основните моменти са в това, че извън зависимостта от типа данни пред тях се поставя ключовата дума const. Освен това константата задължително трябва да бъде инициализирана при създаването си. Да се промени нейното значение след това ще бъде невъзможно.

## Можете да изведете значението на няколко променливи на екрана по следния начин.

Извеждането на значенията на променливите на екрана става с помощта на cout<<

**cout<<име\_на\_променливата; // каизчки в дадения случай не са показани**

Може да се покаже съдържанието на няколко променливи чрез <<

**cout<<имя\_переменной1<<имя\_переменной2;**

**//**

Допустимо е да се редува показването на съдържанието на променливите с текстови съобщения и Escape-последователности чрез <<

**cout<<"Текст"<<имя\_переменной1**

**<<"Текст"<<имя\_переменной2<<"\n";**

Показване на съдържанието на константите се осъществява по аналогия с променливите.

## Практически примери

Ще приведем няколко примера за създаване и инициализация на променливи и константи за различните типове данни.

## Целочислени променливи и константи.

Цели числа ние срещаме навсякъде: възраст, количество столове, количество на стаите, количество на дните в седмицата и т.н.

Променливи, в които ще се съхраняват цели числа, се ОБЯВЯВАТ по следния начин:

int Age;

За какво се говори в този ред? Че в променливата Age(възраст) ще се съхранява цяло значение. Думата int обявява типът на значението на променливата с името Age.

Сега искаме да внесем в променливата Age значение 34. Как да го направим?

Age =34;

Този ред се чете така: «Променливата Age присвоява значение 34».

Още един път ще разгледаме оператора за присвояване: Age =34;

От ляво на знака равно стои името на променливата, на която се присвоява значение. А от дясно стои това значение което се присвоява.

Константа в която ще се съхранява цяло число се обявява по следния начин:

const int Count\_Days\_in\_Week=7;

За какво говори този ред? Думата const подчертава че се обявява константа. Int съобщава че константа ще е цяло число> след това следва името на константата Count\_Days\_in\_Week и нейното значение 7.

Сега да разберем как да се изчисляват значенията на променливите. Защо е нужно това? Прост пример е: как да пресметнем колко часове има 2000-та година? Надали ще искате да пресметнете това число сами.

В интерес на истината е доста лесно да накараме компютъра да направи това самостоятелно. От нас се изисква да напишем само формулата за това изчисление.

В 2000-та година има 366 дни а в тях 24 часа. Съответно формулата за пресмятане на количеството часове през 200-та година е следната: 366 се умножава на 24.

В езика С в качеството на знак за умножение се използва \* (звезда, комбинация от Shift+8)

Да разработим програма която пресмята колко часове има през 2000-та година.

Преди създаването на програмата се съветва кратко да се нахвърли нейния алгоритъм .

Алгоритъм — последователност от действия, насочени за решение на поставена задача.

Дадено: количество дни в годината – 366. Това значение няма да се променя, затова ще го обявим за константа на целия тип с име DayIn\_2000Year. Количеството на часове в деня – 24. Също не се променя. Ще го обявим за константа на този тип с името HourInDay. В нашата програма ще има една променлива, в нея ще запишем резултата на пресмятането. Ще наречем тази променлива HourIn\_Year2000. Тя ще бъде от цялостен тип. (int)

Алгоритъма ще бъде следния:

* 1. Обявяване и инициализация на променливите и константите.
  2. Пресмятане на резултата.
  3. Извеждане на екран резултата.

Имената на променливите можете да измислите сами (не забравяйте само за правилата за съставяне на имената на променливите).

А сега – както винаги, да създадем нов проект и да въведем следния код:

// Заголовок

#include <iostream>

// определение пространства имен, в котором есть cout<< using namespace std;

// Главная функция void main()

{

// вывод пустой строки cout<<"\n";

//Объявляем целочисленные константы int DayIn\_2000Year=366;

int HourInDay=24;

//объявляем целочисленную переменную int HourIn\_Year2000;

// вычисляем искомое значение и

// помещаем его в переменную HourIn\_Year2000 HourIn\_Year2000=DayIn\_2000Year\*HourInDay;

// выводим значение переменной HourIn\_Year2000 на экран cout<<"\t\t In 2000 year "<< HourIn\_Year2000;

cout<<" hours\n ";

}

Това е всичко! Компилирайте програмата!

## Веществени променливи и константи.

Пример за обявяване и инициализация

**float Weight; Weight=12.3452; double weight\_atom;**

**weight\_atom= 2.5194е+017;**

Какво обозначава числото 2.5194е+017?

Това е съкратен запис на веществено число. Нарича се експонирана форма за записване на числата. Представяме ви тайната за разшифроване на написаното. Този набор от символи описва числото 251940000000000000 или 2,1594×1017.

–3.4Е008 се разшифрова така: 3,4×10-8, което е аналогично на 3,4:108.

–1.5E+003 се разшифрова така –1,5×103.

Числа с плаваща точка от типа на float могат да се променят от –3,4×1038 до 3,4×1038.

Значения от –3,4×-38 до 3,4×10–38 се смятат за равни на нулата.

А сега нека да поработим с веществени числа на практика:

Ще напишем програма която ще пресмята стойността на покупката. Нека програмата да използва цената на товара (Cost), количеството на купения товар (Count), и да отчита намалението (Discount), изчислената стойност на покупката (Price). Ще създадем нов проект **Pokupka** и ще въведем следния текст на програмата.

// Заголовок

#include <iostream>

// определение пространства имен, в котором есть cout<< using namespace std;

// Главная функция void main()

{

//Объявляем переменную Discount float Discount=0.05;

//Объявляем переменную Cost float Cost=10.50;

//Объявляем переменную Count int Count=5;

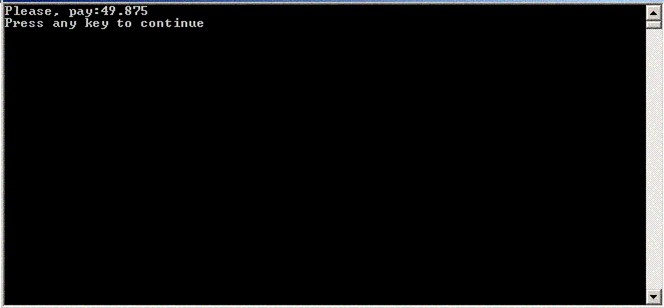
//Объявляем переменную Price float Price;

//Вычисляем значение переменной Price Price=Count\*Cost-Count\*Cost\*Discount;

// Выводим итоговую стоимость товара со скидкой cout<<"Please, pay:"<<Price<<"\n";

}

Компилирайте програмата и я стартирайте. Това което трябва да видите на екрана е показано долу.



## Символни и логически променливи и константи.

В следния урок ние няма да прилагаме примери за използване на символни и логически променливи и константи. Тяхното предназначение по-подробно ще бъде описано по-напред. Ще говорим само за обявяването и инициализацията.

// Логическая переменная bool Flag;

Flag=true;

// Один символ всегда указывается в одинарных кавычках char Symbol='A';

/\* Escape - последовательность рассматривается компилятором, как один символ и соответственно может быть записана в переменную или константу

типа char\*/

const char NewLine='\n';

cout<<NewLine// показывает пустую строку

# Въвеждане на данни

Вие вече сте запознати с операцията за въвеждане на информация на екрана на компютъра – count, но в повечето програми се изисква не само да се извежда каквато и да е информация на екрана, но и да имат възможност да се въведат каквито и да е данни с помощта на клавиатурата. В предишния раздел беше изкарана програма за пресмятане на отстъпки. Естествено е, че параметри от сорта на цена и количество на товара няма да е лошо да бъдат въведени с клавиатурата на етапа на изпълнение на програмата. Нека да разгледаме как можете това да го направите.

Ако на нас ни трябва да въведем данни в компютъра, ще използваме командата cin. Как да я използваме? Синтаксис за въвеждане на операцията:

**cin>>име\_на\_променлива;**

име\_на\_променлива нарежда на променливата в която трябва да се сложат данните въведени чрез клавиатурата:

**Например:**

**cin>>Age;**

Тази команда поставя число въведено от клавиатурата в променливата с име Age. За да се въведе число в променливата Number трябва само да се напише следната команда

**cin>>Number;**

Въвеждане на няколко променливи на веднъж става по следния начин:

**cin>>имя\_пременной1>>имя\_пременной2>>...>>имя\_пременнойN;**

Списъкът с имената на променливите трябва да съдържа имената на всички променливи в които Вие искате да въведете данни чрез клавиатурата. Списъкът на имената може да бъде съставен от всякакво количество имена на променливите разделени чрез комбинация от следните символи >>.

**cin>>Quantity>>Price>>Discount;**

Хайде да добавим в програмата Pokupka въвеждане на данни с клавиатура:

//Заголовок

#include <iostream>

// определение пространства имен, в котором есть cout<< using namespace std;

// Главная функция void main()

{

//Объявляем переменную Discount float Discount=0.05;

//Объявляем переменную Cost float Cost=10.50;

//Приглашение ввести цену товара cout<<"What's the cost?\n";

//Ввод значения в переменную Cost cin>>Cost;

//Объявляем переменную Count int Count=5;

//Приглашение ввести количество cout<<"How much?\n";

// Ввод значения в переменную Count cin>>Count;

//Объявляем переменную Price float Price;

//Вычисляем значение переменной Price

Price=Count\*Cost-Count\*Cost\*Discount;

// Выводим итоговую стоимость товара со скидкой cout<<"Please, pay:"<<Price<<"\n";

}

Сега видяхте особеностите на работа с оператора cin>>. В момента в който програмата срещне даден оператор, тя се спира и очаква реакция на потребителя. И така докато потребителя не въведе данните и не натисне Enter. Само след това ще продължи изпълнението на програмата.

На примера още един път ще поработим с въвеждане и извеждане. Ще напишем програма измамник: програмата предлага игра на числа, който си науми по голямо число, той печели.

Да създадем нов проект **Game** и да въведем следния текст:

// Заголовок

#include <iostream>

// определение пространства имен, в котором есть cout<< using namespace std;

// Главная функция void main()

{

// Приглашение "Давай играть!" cout<<"Let's play!\n";

//Объявление переменной i int i;

//Приглашение "Введите число" cout<<"Enter a number:";

//Ввод числа cin>>i;

//Вывод числа, которое "загадал" компьютер cout<<"I have "<<i+1<<"\n";

// Вывод результата игры cout<<"I'm winner!\n";

}

Компилирайте програмата. Използването на програмата е доста лесно. Просто въвеждате случайно число и постоянно излиза така, че компютъра си е „наумил“ по-голямо число и печели. Ето какво ще видите на екрана при пускане на програмата ако на “Enter a number:” въведете числото 67:

Let's play!

Enter a number: 67 I have 68

I'm winner!

Press any key to continue...

Защо той печели постоянно?

Нека да разгледаме реда

**cout<<"I have "<<i+1<<"\n";**

В този ред е изкарано значението на променливата i, стойността на която сте въвели с клавиатурата и е увеличено с 1, тоест компютъра винаги изкарва число с 1 по-голямо от това което Вие сте въвели.

Ако в тази команда се смени частта i+1 на i-1, то ще печелите винаги вие, защото числото изкарано от компютъра винаги ще бъде с една единица по-малко от това което вие сте въвели.

Като заключение искаме да Ви обърнем вниманието на операторите +(плюс) и – (минус). Те се използват за добавяне и изваждане. В езика С също така има оператор за деление – /. Дадената информация ще ви помогне с домашната работа а по подробно за операторите ще поговорим в следващите уроци.

# Литерали

Литерали (literals) — това са фиксирани значения, които програмата не е в състояние да променя. Във всеки език С съществуват литерали, включващи символни типове, цели числа и числа с плаваща точка.

## Няколко примера:

|  |  |
| --- | --- |
| 5 | целочисленный литерал-int |
| 5l | l или L означает long |
| true | логический литерал-bool |
| 5.0 | литерал с плавающей точкой, понимается как double |
| 5.0f | f или F— с плавающей точкой, понимается как float |
| 0.3е–2 | литерал с плавающей точкой double, e или E отделяют экспоненциалную часть |
| 'd' | символный литерал |
| "Visual" | строкови литерал — това набор произволни сим- волов, заключенный в каизчки. Компилатор вос- принимает его именно как набор символов и никак обрабатывать его не собирается, даже ако в каизчках окажутся какие-то ключевые слова и операции. |

Примеры кода, съдържащего литералы.

//"abrakadabra" - строковый литерал '\n' - символьный литерал cout<<"abrakadabra"<<'\n';

int a = 2; // 2 - литерал типа int

С това представянето на материала от първи урок може да се смята за приключено. Настоятелно препоръчваме да направите домашната работа, описана в следващия раздел. Пожелаваме Ви успех!

# Домашна работа

1. Напишете програма която изчислява средно аритметическото на две числа.
2. Напишете програма която конвертира левове в

$,Є, руски рубли.

1. Въведете на екрана следния текст: "To be or not to be"

\Shakespeare\

1. В C липсва операция возведения в квадрат. Напишете програма, която изчислява квадрат на всяко введен- ного числа.
2. Введите три числа и выведите на екран значение сумы и произведения тезих чисел.

1. Урокът е превод от руски, затова се привежда оригиналният пример. Но той би могъл да бъде актуален и по отношение на българските думи. Макар и не точно същите (бел.преводача) [↑](#footnote-ref-1)