**II. Основни типове данни. Входно изходни операции. Условни оператори. Switch-Case**

1. Структура на програма

**<библиотеки>**

**…деклариране на глобални променливи и спомагателни функции**

**<главна функция>**

**{**

**< тяло на функцията>**

**}**

**<библиотеки>** - в тях се съдържат основни и спомагателни методи, оператори и функции по подразбиране. Библиотеките се изброяват една под друга, като започват **с #include** след което се записва пълен път до файла, който искаме да използваме за библиотека, а ако е от подразбиращите се стандартизирани в езика само името му. Обикновено библиотеките завършват с **.h**, което идва от така наречените **header** файлове – спомагателни с готов за ползване код. Header файлове може да направите и сами, като за използването им е нужно да спазите гореописаните правила.

Когато е от стандартните библиотеки то извикване по следния начин:

**#include <име на библиотека .h>**

Когато е създадена от нас:

**#include “пълен път и име на файла, който ще използваме”**

Между библиотеката и главната функция могат да бъдат декларирани и описвани други функции, които играят спомагателна роля за работата на програмата.

Променливите описани извън функции в основното тяло на програмата се наричат ***глобални***. За тях ще стане дума в следващата точка от упражнението.

**<главна функция>** - главната функция – **main** – е основната функция, от която започва работата на програмата. **Тя е задължителен елемент от програмата и може да съществува само една.** Всяка друга функция играе роля на спомагателна към основната и развива изграждането на програмния код.

**<тяло>** на функция или оператор е мястото, което се огражда с **{ }**, между които пишем код с определена функционалност. Цялото съдържание на кода между скобите се изпълнява при едно логическо използване на оператора или функцията, към които е прикачен. Възможно е да имаме множество вграждания на <тяло> в <тяло>, като се спазва правилото, че скобата за затваряне - } – затваря последното отворено място – {. За да функционира правилно програмата трябва всички тела да са правилно отворени и затворени. **В Python това е реализирано чрез отстъпите навътре.**

Пример за правилно написана програма:

**#include <stdio.h>**

**int main()**

**{**

**printf(“Hello World!\n”);**

**return 0;**

**}**

2. Понятие за променлива

Променливата в програмните езици има за цел да съхранява данни. За да се използва променлива в програмният език С, са необходими две неща: да се избере подходящ тип на данните – това е видът данни, който ще се съхранява в тази променлива, както и име на самата променлива. Създаването на тази променлива се нарича декларация, а записването на информация в нея нарича инциализация. Всяка променлива, използвана в С, задължително трябва да бъде декларирана. По този начин, компилаторът ще знае колко памет да задели и как да интерпретира тези данни.

**Видимост на променлива** – в езика С видимост на променлива или живот означава мястото, на което може да използваме вече декларирана или инициализирана променлива.

***Тук се влияем от няколко правила:***

- Всяка променлива е видима след мястото си на дефиниране/инициализиране, съответно може да бъде ползвана от там нататък в кода.

- Глобалните променливи се декларират извън функциите в общата част на програмата, имат стойност по подразбиране 0 и могат да бъдат ползвани навсякъде в програмата след декларацията им.

- Локалните променливи се използват само в блока код, в който са обявени и нямат стойност по подразбиране по време на декларация. Техният живот е в рамките на обявеното тяло. Локалните променливи са с по-голям приоритет при ползване в кода от глобалните.

***Не могат да съществуват променливи с едни и същи имена и от един и същи вид в едно тяло!***

3. Видове променливи в С

В програмния език С са създадени типове данни, които служат за изграждането на основата за съхранението на данните. Основните типове данни имат за цел да описват цели числа, реални числа и букви. Всеки един тип от тези данни се характеризира със следните параметри – размер байтове(**таблицата е в битове**), които заема в паметта; интервал на стойностите, които могат да бъдат записвани чрез този тип.

Картина, която съдържа текст

Описанието е генерирано автоматично

В таблицата са показани допустимите стойности за всеки един тип за компилатор GCC, разликите се в допустимите стойности се наблюдават, защото стандартът за езика С е гъвкав и има възможност да варират стойностите за различните компилатори.

Картина, която съдържа текст

Описанието е генерирано автоматично

От таблицата се вижда, че числата в плаваща запетая като допустими стойности се различават драстично. Освен това типове като double и long double се препоръчва да се използват, **когато точността на реалните числа е от значение**.

*Типовете данни, които описват реалните числа се наричат „числа с плаваща запетая“, защото в представянето на числата в компютъра се извършва използвайки записване чрез мантиса и порядък, които се записват на различни места в двоичния запис на числото*

**Типове от данни описващи букви**

В програмният език С съществува един единствен тип данни, който описва букви и символи. Този тип е char (съкратено от character от англ. „буква, знак“). Този тип както беше споменато е и тип, описващ целочислени числа. Причината типът да има така дуалност (двойнствен характер) се крие в представянето на символите в една компютърна система. Всички възможно символи са записани в една таблица (ASCII таблица), в която срещу всеки символ е записано съответното му число. Така, когато трябва стойността на променливата от тип char да бъде представена като буква, компютърът автоматично заменя числовата ѝ стойност, със символа записан срещу нея. По този начин символите като ‘A’ имат стойност 65, а ако към буквата прибавим число получим символ, който седи на отстояние толкова места в таблицата, колкото е стойността на числото.

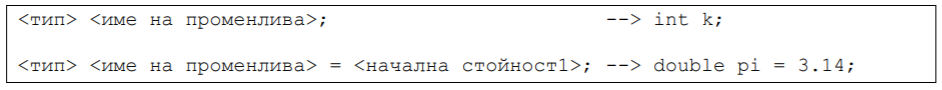
**Безнакови типове данни**

В програмният език С има типове, които описват само цели положителни числа. Тези типове данни се получават като пред целочислените типове данни се запише unsigned. По този начин новите типове имат следните допустими стойности:

Картина, която съдържа текст

Описанието е генерирано автоматично

4. Създаване и основни операции с променливи

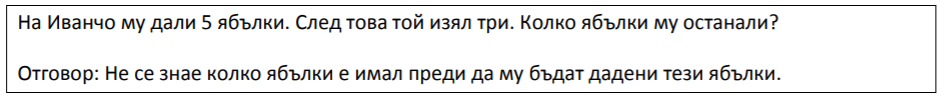
Създаване на променливи/деклариране на променливи се извършва в началото на тялото на всяка функция, според стандарта на С. За да бъде създадена една променлива е нужно да се следва един от следните два модела:

В двата модела се заместват полетата както следва:

**<тип>** - това поле се замества с желаният тип, който е нужен за съхраняване на данните(int, char, float…); **В езикът Python отново имаме същата концепция с типа данни, но тя се реализира имплицитно от езика и е скрита за потребителя.**

**<име на променлива>** - това е името, което ще се използва за достъп то тези данни;

**<начална стойност>** - може да се зададе начална стойност на дадената променлива(декларация + инициализация).

Важно е да се отбележи, че в програмният език С компилаторът не се грижи за зануляване на стойностите при тяхното дефиниране. Тоест, ако имаме локална променлива **int p** и не сме иницализирали **p** то тя може да има стойност 0 или нещо друго. Добра практика да се задава начална стойност. Този проблем може да се демонстрира и със следната логическа задача:

На един ред може да има дефинирана повече от една променлива от един и същи тип. Имената на променливите се разделят със запетая.

**#include <stdio.h>**

**int main()**

**{**

**Int number, number2, number3;**

**return 0;**

**}**

5. Операции с елементарен тип данни

Операциите с елементарните типове данни се разделят на няколко типа - аритметични, логически и побитови. Побитовите операции са само за целочислени типове и могат да бъдат разгледани само по ваше желание.

Картина, която съдържа текст

Описанието е генерирано автоматично**Аритметични операции:**

Картина, която съдържа текст

Описанието е генерирано автоматично

6. Логически оператори в езика С

Логическите операции са част от операциите, с които може да се работи с елементарните типове данни.

В програмният език С под „истина“ се разбира ненулева стойност, а под „лъжа“ – нула.

**Картина, която съдържа текст

Описанието е генерирано автоматичноВидове логически операции:**

Картина, която съдържа текст

Описанието е генерирано автоматично**Сравнения:**

**Различните видове логически операции и сравнения могат да бъдат обединявани по различен начин в зависимост от комплексността на казуса, който решаваме.**

* **(a > b) && (b > c) това ще върне „истина“, ако и двете условя са правилни**
* **(a > b) || (b > c) това ще върне „истина“, ако едно от условията е вярно**

**\* Може да потърсите „Таблица на истинност“ за да си разясните по-подробно как работят логическото „и“ и логическото „или“.**

**7. Вход и изход**

Служат за комуникация с потребителите. Под вход обикновено се разбира клавиатурата, а под изход - екрана.

**А) Вход** - въвеждане на данни в програмата (текст, параметри) от потребителя

**Б) Изход** - извеждане на данни в разбираем за потребителя вид

И двете използват т.нар. форматиращ низ за преобразуване на данните в последователност от символи и обратно. Може да съдържа текст и форматни определители - специални символни комбинации, задаващи правилата за извеждане/въвеждане на данните. Примери са дадени в следващата таблица.

|  |  |
| --- | --- |
| **Символ** | **Значение** |
| \a | Издаване на звук |
| \n | Нов ред |
| \t | Табулация |
| \r | Връщане в началото на реда |
| %d | int |
| %lf | double |
| %f | float |
| %c | char |
| %s | string |
| %hi | short |

**Това са само част от така наречените форматни спецификатори и специални символи, ако се нуждаете от други просто потърсете в гугъл.**

**В) Употреба**

Най-често се прилагат чрез функциите printf() и scanf(). **Алтернативи в Python – print и input.**

Функцията printf() е основната функция за изход в езика C. Тя има следния формат:

printf(<форматиращ\_низ>, <променлива\_1>, <променлива\_2>,...);

където <променлива\_1>, <променлива\_2> - имената на променливите, чиито стойности трябва да се изведат на екрана. printf не е задължително да получава някакви променливи

Пример: printf("Стойността на x е %d.", x); - > на мястото на %d в изхода ще се изведе стойността на променливата x декларирана някъде в програмата.

Функцията scanf е основна функция за вход в езика C и има следния формат:

scanf(<форматиращ\_низ>, <местоположение\_на\_променлива\_1>, <местоположение\_на\_променлива\_2>,...)

<местоположение\_на\_променлива\_1>, <местоположение\_на\_променлива\_2,...> - описват местата на променливите (чрез оператора '&'), където могат да се запишат прочетените данни. &x се чете като "адреса на x". **За сега запомнете, че като използвате scanf() и подавате променливи пред всяка трябва да има &.** Разбира се по-напред в упражненията ще разберем, защо това е така и в кои случай не се поставя.

Пример(**псевдопрограма**):

Int a, b;

scanf("%d %d", &a, &b);

<въвеждаме готинки числа: 1 2>

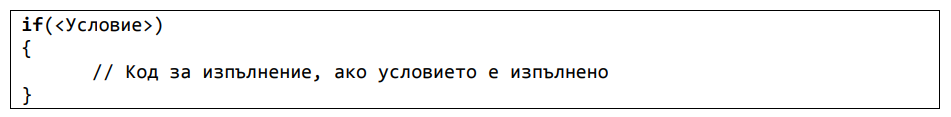
printf("%d + %d = 3", a, b)

На конзолата ще се изведе: 1+2 = 3

**И двете функции по-горе връщат някаква целочислена стойност освен основното им предназначение. За това ще стане дума в упражнението за функции.**

**8. Оператор if**

В повечето програмни езици присъства възможност за разклоняване логиката на програмата – възможността да се изпълняват дадена част от кода само при изпълнено дадено условие, ако това условие не е изпълнено, кодът не се изпълнява. В програмният език С тази възможност се реализира чрез оператора if. Употребата се извършва по начин, показан по-долу:



Под ***<условие>*** се разбира логическо условие, което може да бъде „истина“ или „лъжа“. Ако си спомняте по-горе говорихме точно за логически условия, които ни „казват“ дали нещо е истина или лъжа. Точно тук идва тяхната употреба. Трябва да се отбележи, че операторът if работи и с числа тоест ще приема за истина всяко число различно от 0. Самата истина всъщност представлява точно това -1, лъжата - 0. Така е имплементиран, така наречения Boolean. Тоест нищо не ни пречи да напишем нещо такова:

if (4)

{

<Тяло> //тук тялото винаги ще се изпълни

}

По-нормална употреба:

if (a > b)

{

<Тяло>

}

**9.Оператор if-else**

Употребата на този оператор е идентична на оператор if, но тук е предвиден код, който може да се изпълни само, когато условието не е изпълнено.

Картина, която съдържа текст

Описанието е генерирано автоматично

По този начин програмата има различно поведение в зависимост от истинността на поставеното условие.

**Нищо не ни пречи и да имаме повече от едно условие!**

if (<условие1>)

{

<Тяло>

}

else if (<условие2>) **Алтернативата на това в Python е elif**

{

<Тяло>

}

else

{

<Тяло>

}

**Тук проверяваме условие 1, ако то е вярно влизаме в неговото тяло, ако не е проверяваме условие 2 и пак, ако е вярно влизаме там, ако не е отиваме към „общия“ случай.**

**10. Оператор switch-case В Python този оператор не съществува**

Идеята на този оператор е подобна на if, но тук има повече от две възможни стойности за тестване, като за всяка възможна стойност (случай) се изпълняват различни част от кода.

**Употреба:**

Картина, която съдържа текст

Описанието е генерирано автоматично

Значение на елементите:

***<променлива>*** - тук се записва името на променливата, чиято стойност ще бъде изследвана. Типът на променлива трябва да бъде задължително изброим тип (например цяло число или буква).

***<стойност1>, <стойност2>…*** – константи, които показват случаите, за които ще се изследва променливата.

***case*** – запазена дума, която показва от къде започва изпълнението на кода, който се отнася за определената стойност. Изпълнението продължава до срещане на запазената дума break, възможно е break да се намира в следващ случай (както е показано в примера), така ако променливата има стойност 1, то ще се изпълни „Код за изпълнение 1“ и „Код за изпълнение 2“.

***default*** – в този случай се влиза, ако стойността на променливата не е описана в предходните случаи.

**Нищо не ни пречи вместо switch/case да използваме if/else с много проверки, но switch/case е по-бърз в повечето случаи!**

**Задачи за точки:**

1. Като използвате printf и scanf, въведете две числа и ги изведете в конзолата.
2. Като използвате if-else, проверете дали първото число е по-голямо от 3 и дали второто е по-малко от -1, ако това е вярно, изведете на конзолата: „Yes yes dob dob“. Ако първото е 0, а второто е 5, изведете: “W W W W”.

Ако никое от тези условия не е вярно, изведете: “Sry”.

1. Като използвате switch-case, проверете дали първото число е 1, ако е, изведете стойността на второто число + 1, ако е 2, изведете второто число + 2, ако е 3 изведете второто число + 3, ако никое не е вярно, изведете второто число + 10. **Използвайте форматиращ низ за извеждане!**

**Още задачи: :DD**

1. Да се напише програма, която намира лице на правоъгълник по зададени от клавиатурата две страни и го извежда на екрана.
2. Да се напише програма, която намира обиколката на окръжност по зададен диаметър от клавиатурата и извежда резултата на екрана.
3. Да се прочете от клавиатурата едно целочислено число и да се изведе съобщение, ако то е по-голямо от 6.
4. Да се прочете от клавиатурата едно целочислено число и да се изведе на екрана съобщение, ако остатъкът от делението му с 8 е по-голям от 4.
5. Да се прочетат от клавиатурата 3 числа и да се изведе на екрана най-малкото от тях.
6. Да се прочете от клавиатурата цифра и да се изведе на екрана нейното име. Да се използва оператор switch-case.
7. Да се прочете от клавиатурата едно число от 1 до 7 и да се изведе на екрана кой ден от седмицата съответства на това число. Да се предвиди случай, когато не е въведено валидно число.