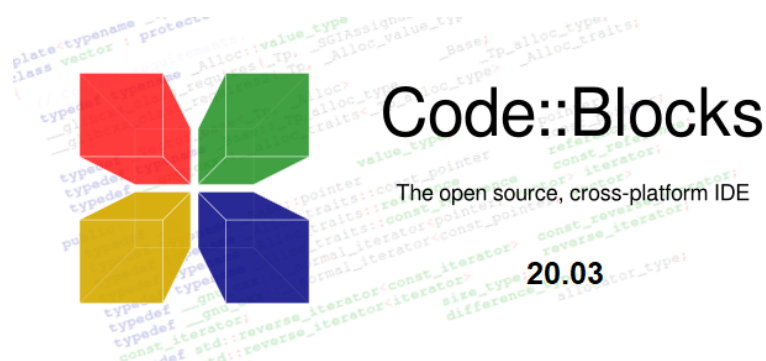


I. Запознаване със средата за разработване на програми на C. Компилиране, настройка и изпълнение на програма, стандартен вход/изход.

За работа с езика C съществуват множество методи и среди за писане. Самият език често е вграден като компилатор в Операционната система, която използваме. Самите компилатори от езика към машинен код са няколко вида, всеки със своите особености и разновидности, но в общия случай ние ще разгледаме основните елементи за изграждането на програма и проследяването на логика чрез разбиването на сложна задача към опростени стъпки и логическа последователност. В конкретния случай използваме MinGW пакета със съдържащият се вътре компилатор GCC. За среда за разработка препоръчваме лесно усвоимо IDE, което е безплатно и леко за работа на всяка машина.

1. Code::Blocks

Основната цел на една интегрирана среда за разработка (**Integrated Development Environment**) е осигуряване на условия за по-лесно изграждане на софтуер.



Такъв пример е средата Code::Blocks, предоставяща множество удобства в тази насока, а именно:

- **Компилатор** - програма, "превеждаща" (компилираща) език от високо ниво (разбираем за хората) в машинен (разбираем за компютъра).
- **Редактор на код** - поле, в което се пишат инструкциите на програмата. Използва следните инструменти за по-добра четимост:
 - **синтактично подчертаване (Syntax High-Lighting)** - оцветяване на ключови думи според ролята им в програмата.
 - **сгъване на код (Code Folding)** - скриване на определена част от кода (например тяло на функция)

- **допълване на код (Autocomplete)** - подсказване за наименование, ключова дума и т.н.
- **Дебъгер** - проследява процеса на изпълнение на програмата. Чрез него се търсят грешки ("бъгове") в логиката ѝ.

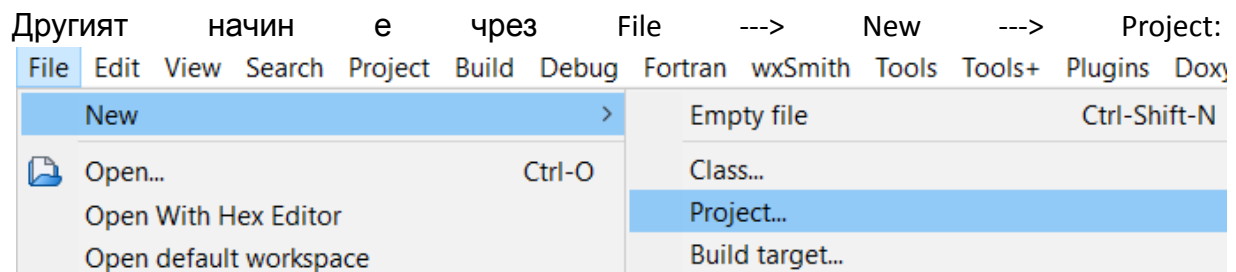
Линк за сваляне на средата за работа:

- <https://www.fosshub.com/Code-Blocks.html?dw=codeblocks-20.03mingw-setup.exe> или
- <http://sourceforge.net/projects/codeblocks/files/Binaries/20.03/Windows/codeblocks-20.03mingw-setup.exe>

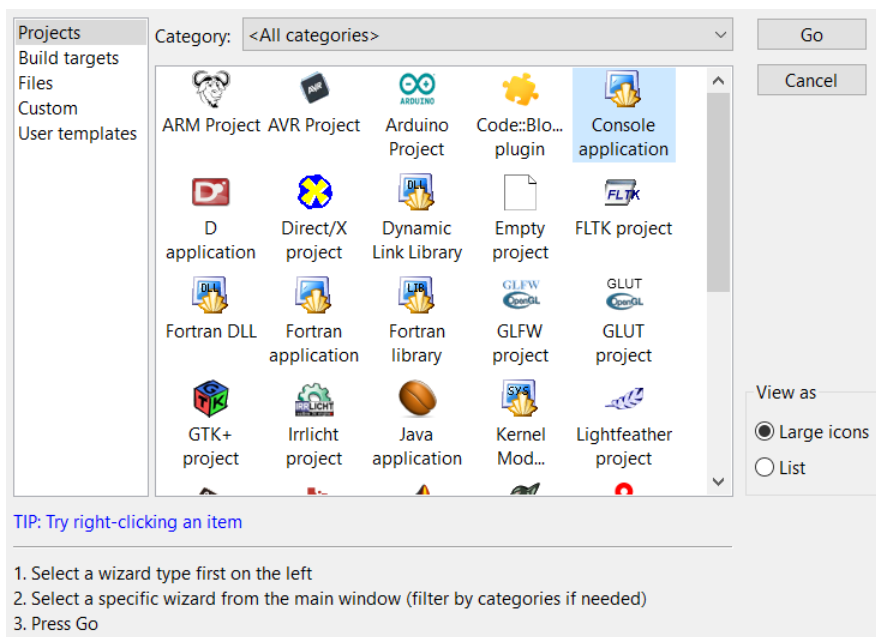
2. Първи проект в Code::Blocks

А) Създаване

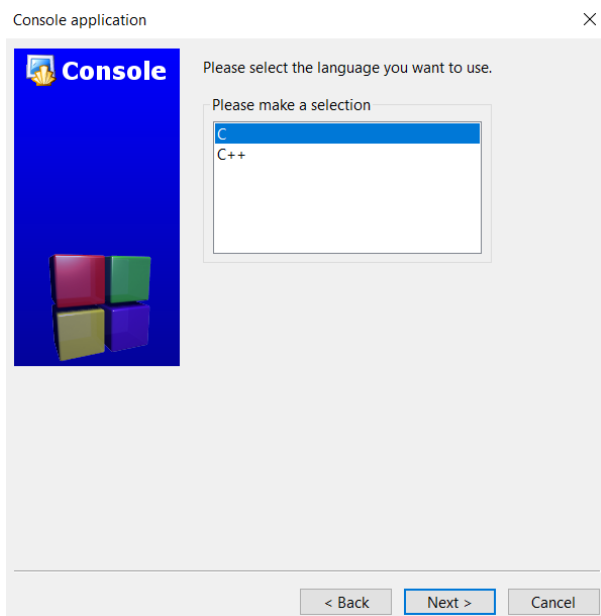
След отваряне на средата, се вижда следната иконка на началната страница:



Избира се "Console Application" и се натиска бутонът "Go":



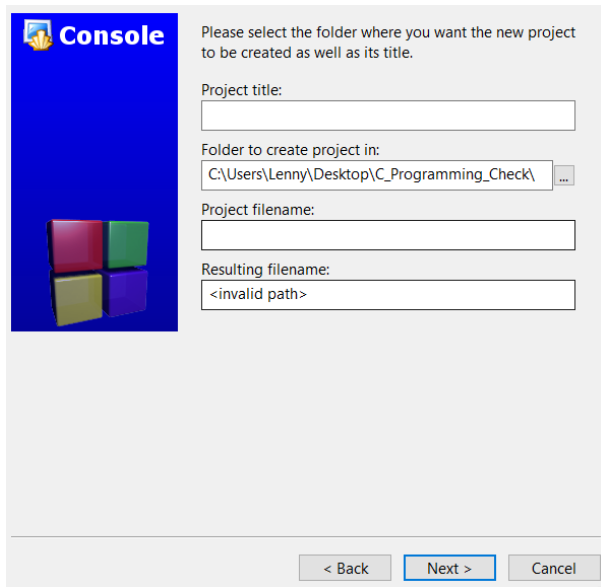
За избор на език, се натиска "C":



Следващият прозорец извежда полета, в които трябва да се попълнят:

- Заглавието на проекта
- Папката (местоположението) на проекта

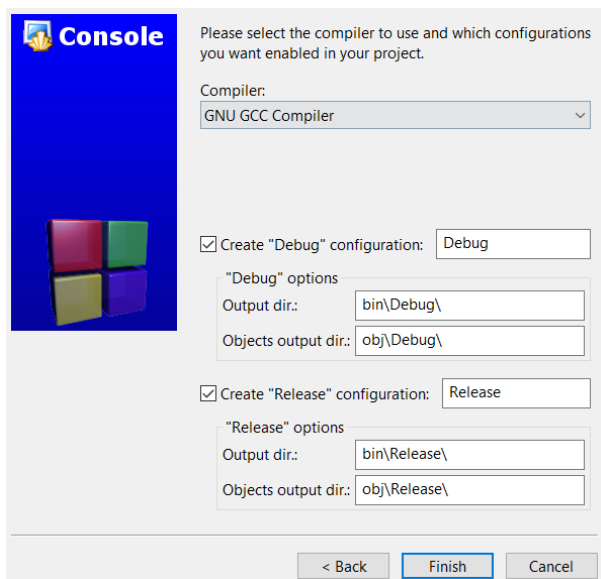
Останалите две полета се попълват автоматично.



The screenshot shows a dialog box titled "Console" with a blue sidebar containing a Windows logo. The main area has a light gray background and contains the following fields and controls:

- Text: "Please select the folder where you want the new project to be created as well as its title."
- Field: "Project title:" with an empty text box.
- Field: "Folder to create project in:" with a text box containing "C:\Users\Lenny\Desktop\C_Programming_Check\" and a browse button "...".
- Field: "Project filename:" with an empty text box.
- Field: "Resulting filename:" with a text box containing "<invalid path>".
- Buttons at the bottom: "< Back", "Next >" (highlighted with a blue border), and "Cancel".

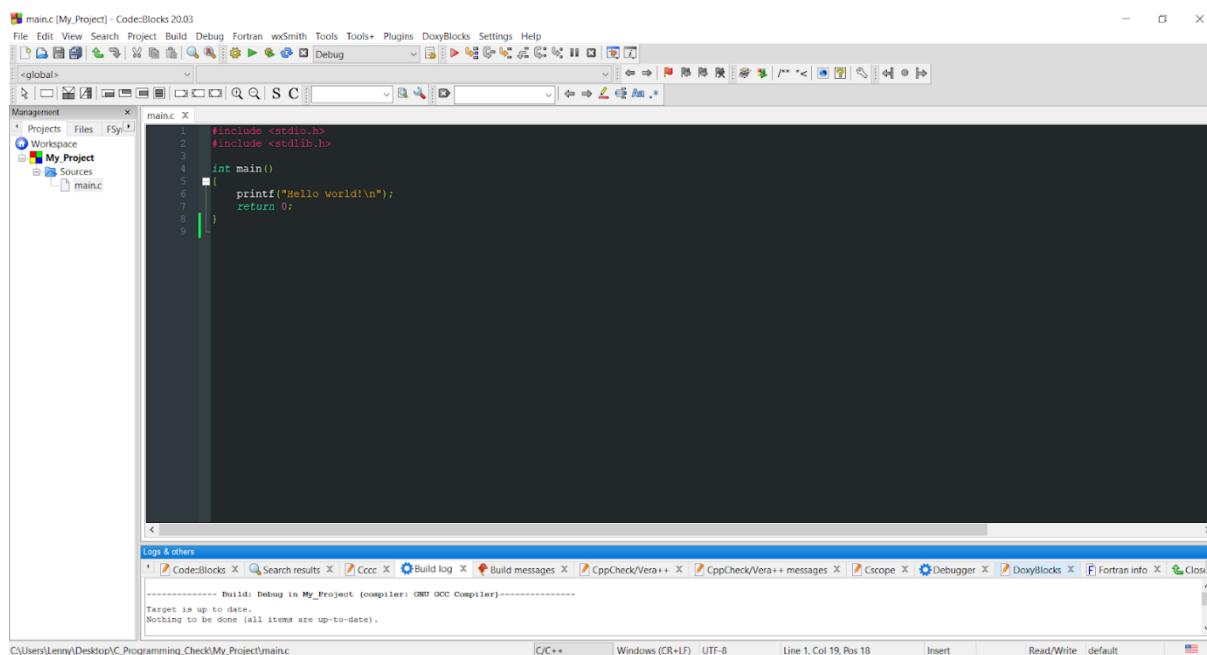
Следва видът компилатор, който ще се използва (обикновено GCC), както и "Debug" и "Release" настройки. Не подлежат на промяна.



The screenshot shows a dialog box titled "Console" with a blue sidebar containing a Windows logo. The main area has a light gray background and contains the following fields and controls:

- Text: "Please select the compiler to use and which configurations you want enabled in your project."
- Field: "Compiler:" with a dropdown menu showing "GNU GCC Compiler".
- Field: "Create 'Debug' configuration:" with a checked checkbox and a text box containing "Debug".
- Field: "'Debug' options" with a sub-label and two text boxes: "Output dir.:" containing "bin\Debug\" and "Objects output dir.:" containing "obj\Debug\".
- Field: "Create 'Release' configuration:" with a checked checkbox and a text box containing "Release".
- Field: "'Release' options" with a sub-label and two text boxes: "Output dir.:" containing "bin\Release\" and "Objects output dir.:" containing "obj\Release\".
- Buttons at the bottom: "< Back", "Finish" (highlighted with a blue border), and "Cancel".

След натискане на "Finish", се появява прозорецът за разработка на проекта:



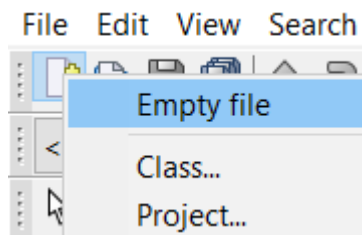
Б) Добавяне на файлове

- Добавяне на празен файл

Чрез File ---> New ---> Empty File:

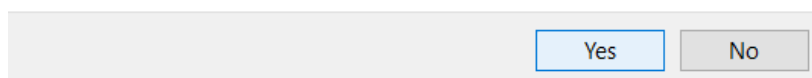
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins Doxyf		
New	>	Empty file Ctrl-Shift-N
Open...	Ctrl-O	Class...
Open With Hex Editor		Project...
Open default workspace		Build target...

Другият начин е чрез иконката с плюското:

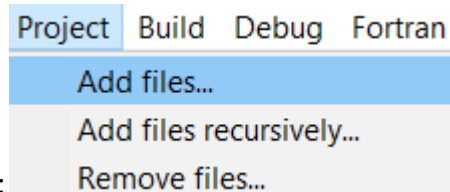


"Add File To Active Project" запазва файла в "source" папката на проекта:
Add file to project

Do you want to add this new file in the active project (has to be saved first)?



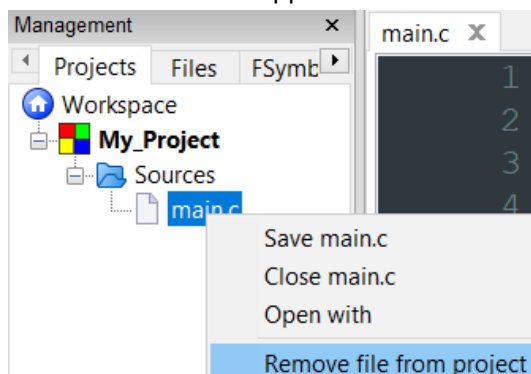
Добавяне на съществуващ файл



Чрез Project ---> Add Files:

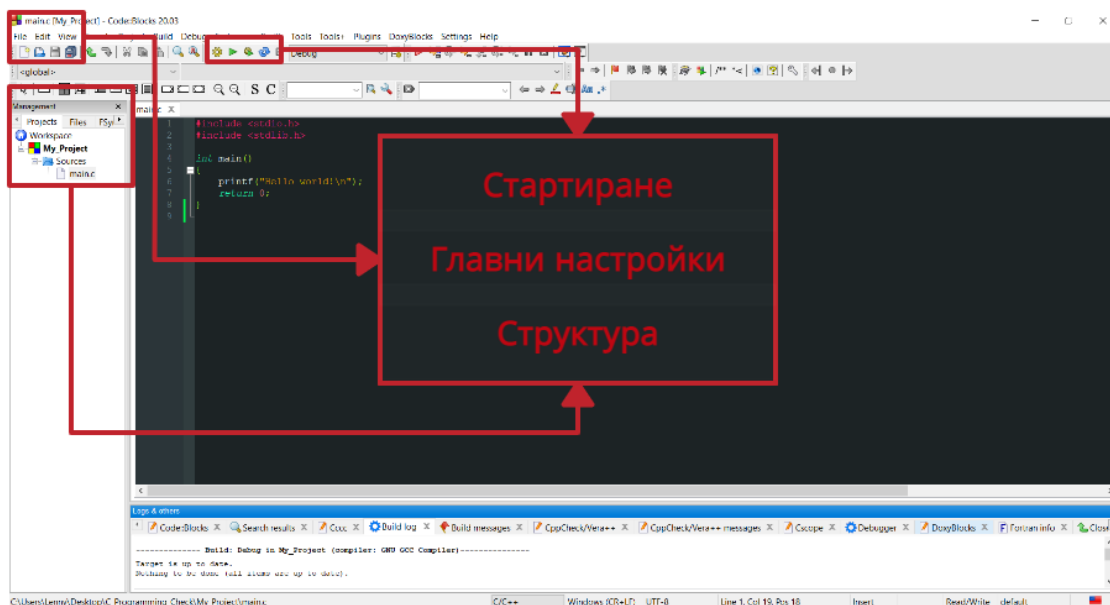
В) Премахване на файл

Натиска се с дясно копче и се избира "Remove File From Project":



Това не изтрива физически файла, а само го премахва от проекта.

3. Основни компоненти на средата



А) Структура

Описана в полето "Management", тя демонстрира йерархичната подредба на всички файлове в проекта. "main" файлът се създава по подразбиране в началото на всеки проект и служи за начална точка на програмата.

Б) Главни настройки

Служат за добавяне на файлове (демонстрирано горе), настройки и запазване на промените в проекта.

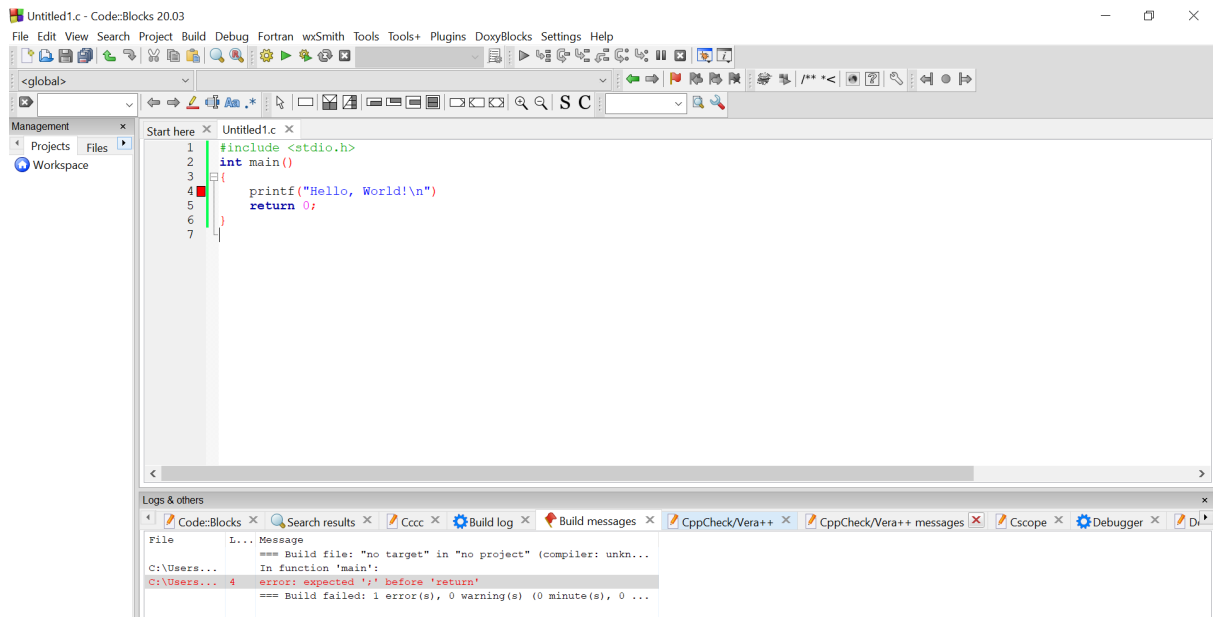
В) Стартиране

Ролята на тези бутони е първоначална компилация и стартиране на проекта.

- Опцията **Build** компилира кода до работен файл за изпълнение.
- Опцията **RUN** изпълнява последно успешно построеният код.
- Третата опция компилира кода и го изпълнява.

При допускане на грешка в синтаксиса на програмата средата подава къде е намерена първата грешка, на кой ред е и в какво се състои, като най-долу в средата е наличен прозорец с обяснения. Дава се и маркер на реда в самия код където е открило проблем при компилация. При зареждане а голямо количество код и наличие на множество грешки, не е гаранция, че последващите грешки са

такива до отстраняване на първата налична. В някои случаи средата може да се бърка от тази първа грешки и да не асемблира правилно остатъка от кода.



4. Структура на програма

```
#include <stdio.h>
int main()
{
    printf("Hello, World!\n");
    return 0;
}
```

Този код цели да провери дали средата е правилно настроена и се компилира правилно. Състои се от най-важните и минимални изисквания, нужни за работата на програмата:

#include <stdio.h> - библиотека, включваща предварително описани функции, в случая за работа със стандартните вход (клавиатура) и изход (екран) на компютъра

int main() {} - главна функция. Всичко, написано в тялото ѝ, се изпълнява, когато програмата започне. Това е входната точка при изпълнението, от нея се извикват всички останали функции.

В една програма може да имате **САМО един MAIN** метод!!!

Честа грешка при начинаещи програмисти е създаването на проект и добавянето на няколко файла, с цел да се запишат решенията за всички поставени задачи на едно място. В този случай се пишат няколко main метода и средата не може да започне работа, защото не знае кой метод всъщност е главният. За целта на курса препоръчваме да не създавате проекти, а да правите само нови файлове от иконката с файл с плусче. По този начин ако нямате активен проект

първоначално няма да асоциирате новия файл с предишните и няма да срещнете проблема с множеството `main` методи.

return 0; - показва, че програмата е прекратила работата си нормално.

printf("Hello, World\n"); - функция за извеждане на текст на стандартния изход (екрана).

5. Стандартни вход и изход

Служат за комуникация с потребителите. Както беше споменато горе, 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.

Форматиращият низ може да включва специални символи за по-голяма четимост на резултата (обикновено разположени накрая).

Символ	Значение
\a	Издаване на звук
\n	Нов ред
\t	Табулация
\r	Връщане в началото на реда

В) Употреба

Най-често се прилагат чрез функциите `printf()` и `scanf()`.

- `printf(<форматиращ_низ>, <променлива_1>, <променлива_2>, ...);`

`<променлива_1>`, `<променлива_2>` - имената на променливите, чиито стойности трябва да се изведат на екрана

Пример: `printf("Стойността на x е %d.", x);` // `x` е името на използваната променлива

- `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".

Пример: <code>scanf("%d %d", &a, &b);</code>
--

Задачи за работа в час:

1. Инсталирайте средата и се запознайте с възможностите ѝ, създайте проект и файл за работа. Тествайте работата на средата с примерния код.
2. Чрез операторите за изход изкарайте на екрана съобщения, като по този начин усвоите възможностите на форматиращият низ.

За помощ в работа с езика C може да се обръщате към официалната документация:

cplusplus.com - [The C++ Resources Network](http://cplusplus.com)

(макар да е за езика C++ съдържа и целият набор от възможности за C)