Универзитет „Св. Кирил и Методиј“ - Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Аудиториски вежби 1

Вовед во C

Околини за развој (IDE)

Структурно програмирање

Содржина

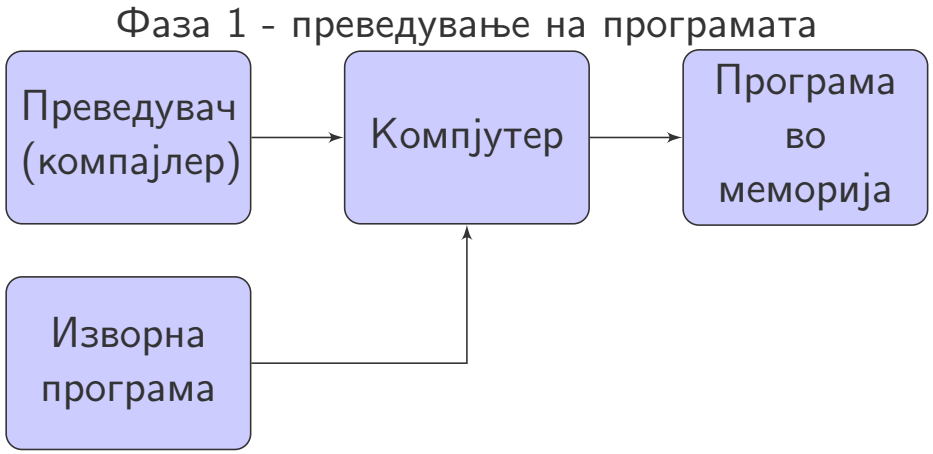
Програмирање (1)

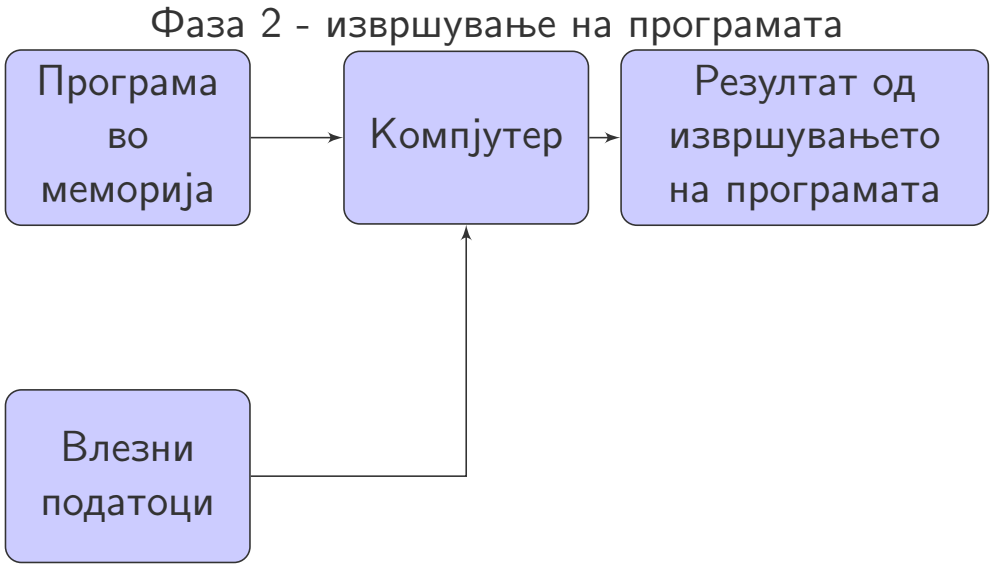
- Програмите што се извршуваат на компјутерот се последователност од **нули и единици** и тоа е единствениот јазик што компјутерот го разбира
- Програмерите пишуваат програми со помош на јазици кои се разбирливи за нив, односно **јазици за програмирање**
- Програма напишана во јазик за програмирање ја нарекуваме **изворна програма**

Програмирање (2)

- За пишување програми често се користат **околини за развој**
- Програмата се внесува преку текстуален уредувач
- Потоа се врши преведување на програмата (**компајлирање**)
- Со тоа се создава **извршна програма** т.е. програма напишана во јазикот на компјутерот

Тек на преведување и извршување





Вовед во програмскиот јазик C

- Развиен во лабораториите на Bell во периодот од 1969 од 1973 од страна на Dennis Ritchie
- Еден од најшироко употребуваните јазици за програмирање со општа намена на сите времиња
- Има огромно влијание во создавањето на многу други јазици за програмирање
  - C++
  - Objective C
  - PHP
  - Java

Синтакса на C

Азбуката е множество на следните дозволени симболи:

a-z, A-Z, 0-9 и ~!@#$%^&*()-+={}[]:;'"<>?/._

Внимание!

Компајлерот разликува големи и мали букви!

Од азбуката на C се формираат зборови кои може да бидат:

1. Клучни зборови
2. Бројни и симболички константи
3. Идентификатори
4. Стрингови (низи од знаци)
5. Оператори

Синтакса на C

Множество на клучни зборови (32)

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
go	if	static	while

Структура на програма во C

Севкупниот изворен код кој се пишува во програмскиот јазик C е организиран во функции

Програма во C

```
int main() {
    deklaracija_na_promenlivi;
    programski_naredbi;
}
```

Програма во Паскал

```
Program ime_na_programata;
var deklaracija_na_promenlivi;
begin
    programski_naredbi;
end.
```

Функции во C

main

Главна фунција во C

()

Во мали загради се примаат влезните аргументи

int

Видот на податокот кој се враќа како резултат стои пред името на функцијата

{}

Телото на функцијата започнува со {, а завршува со }

;

Сите наредби се одделуваат меѓусебно со ;

Употреба на коментари

- За дополнително до објаснување или документирање на изворниот код се користат коментари
- Во C постојат два видови на коментари
  1. коментари во еден ред
  2. коментари во повеќе редови

1. Коментар во еден ред

```
// komentar vo eden red
```

2. Коментар во повеќе редови

```
/* Komentar
   vo povekje redovi */
```

Примери

Пример 1

```
#include <stdio.h>
// glavna funckija
int main() {
    /* funkcija za pecatenje na ekran */
    printf("Dobredojdovte na FINKI!\n");
    return 0;
}
```

Структура на програма во C (проширена)

INCLUDE секција	содржи `#include` изрази за вклучување на надворешни библиотеки, односно користење веќе дефинирани надворешни функции
DEFINE секција	содржи `#define` изрази за декларирање на константи и податочни типови
...	дефинирање на глобални променливи и функции
`int main()`	главна функција

Претпроцесор

- Во C преведувањето (компајлирањето) на програмите го извршуваат:
  - претпроцесорот
  - компајлерот
- Претпроцесорот се управува со помош на т.н. директиви
  - Секоја директива започнува со #

Датотека со заглавја

- Една примена на претпроцесорски наредби е вклучување на „датотека со заглавија“ (анг. header file)
  - Се користи за декларација на функции и променливи на одредена предефинирана библиотека
  - Корисниците ја вклучуваат „датотеката за заглавија“ со цел да ги користат функциите и надворешните променливи
- Вклучување се врши со претпроцесорската директива `#include`
  - Наредбата `#include` предизвикува копија од дадена датотека да се вклучи на местото каде што е испишана директивата

Форми на include

- Има две форми на оваа директива:
  - датотеката која се вклучува може да биде ставена во наводници(""),
  - или во аголни загради (<>)

Пример

```
#include <imedatoteka.h>
#include "imedatoteka.h"
```

- Разликата е во локацијата во која препроцесорот ја бара датотеката која треба да ја вклучиме
  - Со аголни загради (се користат за датотеки од стандардните библиотеки)
  - Со наводници (препроцесорот прво ја бара датотеката во истиот директориум каде што се наоѓа C датотеката која треба да се компајлира)

Променливи (variables)

- Променливите се симболички имиња за места во меморијата во кои се чуваат некакви вредности
- Сите променливи пред да се користат треба да се *декларираат*
- Со секое ново сместување на вредност во променливата, старата вредност се брише

Начин на декларација на променливи:

Видови на променливи

Видови на променливи во C

Цели броеви	Знаковни	Децимални
`int`	`char`	`float`
`short`		`double`
`long`		

Дефинирање на имиња на променливи

- При именувањето на променливите може да се користат:
  - мали букви од a до z;
  - големи букви од A до Z;
  - цифри од 0 до 9 (не смее да започнува со цифра);
  - знак за подвлекување _ кој се третира како буква (не е препорачливо да започнува со _);

Треба да се внимава!

- најчесто должината на имињата на променливите е до 32 знаци
- C ги разликува малите и големите букви!

Примери

Пример 2

```
#include <stdio.h>

int main() {
    int a, b, c;
    a = 5;
    b = 10;
    c = a + b;
    return 0;
}
```

Константи

- Со помош на константи се означуваат вредности кои не се менуваат во текот на извршувањето на програмата
- Секоја константа припаѓа на некој од податочните видови
- Во C постојат неколку типови на константи:
  - децимални: `1, -23, 15`
  - октални: `015, 035, 0205`
  - хексадецимални: `0x25, 0xA4C`
  - реални: `3.5F, -2.845F, 1.34e-9`
  - знаковни: `'a', '_', 'e'`
  - текстуални: `" ", "Koncepti za razvoj na softver"`

Одредување на типот на константите

- Одредувањето на типот на променливите е едноставно (се гледа од самата декларација на променливата)
- Константите не се декларираат и нивниот тип се одредува преку начинот на кој се напишани:
  - Броевите кои содржат "." или "e" се `double`: `3.5, 1e-7, -1.29e15`
  - За наместо double да се користат `float` константи на крајот се додава "F": `3.5F, 1e-7F`
  - За `long double` константи се додава "L": `1.29e15L, 1e-7L`
  - Броевите без ".", "e" или "F" се `int`: `1000, -35`
  - За `long int` константи се додава "L": `9000000L`

Именувани константи (1)

Именуваните константи се креираат со користење на клучниот збор `const`

Пример 3

```
#include <stdio.h>

int main() {
    const long double pi = 3.141592653590L;
    const int denovi_vo_nedelata = 7;
    const nedela = 0; /* po default int */
    denovi_vo_nedelata = 1; /* greshka */
}
```

Именувани константи (2)

Именуваните константи може да се креираат и со користење на претпроцесорот и за нив по правило се користат големи букви

Пример 3

```
#include <stdio.h>
#define PI 3.141592653590L
#define DENOVI_VO_NEDELATA 7
#define NEDELA 7
int main() {
    long double pi = PI;
    int den = NEDELA;
}
```

Оператори

- Операторите се користат за градење на изрази, при што операциите се изведуваат од лево надесно со што се применува правилото на приоритет на операторите во нивното изведување
- Постојат три видови на оператори
  - Аритметички оператори
  - Релациони оператори
  - Логички оператори

Аритметички оператори

Се применуваат на броеви (цели или децимални)

Оператор	Операција
+	Собирање
-	Одземање
*	Множење
/	Делење
%	Делење по модул

Релациони оператори

Се применуваат над било кои споредливи типови на податоци, а резултатот е цел број 0 (неточно) или 1 (точно).

Оператор	Значење
<	Помало
<=	Помало еднакво
>	поголемо
>=	поголемо еднакво
==	еднакво
!=	различно

Логички оператори

Се користат најчесто во комбинација со релационите оператори за формирање на сложени логички изрази, кои повторно враќаат резултат 0 или 1

Оператор	Операција
&&	Логичко И
||	Логичко ИЛИ
!	Негација

Дополнителни оператори

- Оператор за доделување =
- Оператори за инкрементирање и декрементирање

  `++, --`
- Користење на операторите + и − на унарен начин

```
X = + Y;
X = - Y;
```

- Двојни оператори
  - Комбинација од оператор за доделување и друг оператор

    `(+=, -=, *=, /=, %=)`

Оператор за доделување =

- Сите изрази имаат вредност, дури и оние кои содржат =
- Вредноста на таков израз е вредноста на изразот кој се наоѓа на десна страна
- Затоа е можно и доделување од следниот облик:

```
x = (y = 10) * (z = 5);
x = y = z = 20;
```

Двојни оператори

Оператор +=

```
a += 5; // a = a + 5;
a += b * c; // a = a + b * c;
```

Оператор -=

```
a -= 3; // a = a - 3;
```

Оператор *=

```
a *= 3; // a = a * 3;
```

Оператор /=

```
a /= 3; // a = a / 3;
```

Оператор %=

```
a %= 3; // a = a % 3;
```

Работа со променливи и оператори

Пример 4

```c
#include <stdio.h>
int main() {
    int a;
    float p;
    p = 1.0 / 2.0; /* p = 0.5 */
    a = 5 / 2;     /* a = 2   */
    p = 1 / 2 + 1 / 8; /* p = 0; */
    p = 3.5 / 2.8; /* p = 1.25 */
    a = p; /* a = 1 */
    a = a + 1; /* a = 2; */
    return 0;
}
```

Печатење на стандарден излез

- Во C не постои наредба за печатење на екран
- Се користи готова функција од библиотеката за стандарден влез и излез `stdio.h` (**st**and**a**r**d** **i**nput/**o**utput)
  `#include <stdio.h>`
- Функцијата која се употребува е:

```
int printf(kontrolna_niza, lista_na_argumenti)
```

- Контролната низа содржи било каков текст, ознаки за форматот на печатење на аргументите предводени со % или специјални знаци кои започнуваат со \.
- Ознаките за форматот на печатење се одредуваат според видот на променливата чија вредноста треба да се испише.

Ознаки за форматот на печатење

Ознака	Објаснување
%d	за цели броеви
%i	за цели броеви
%c	за знаци
%s	за низа од знаци
%e	реален број во технички формат (e)
%E	реален број во технички формат (E)
%d	реален број во децимален формат
%f	реален број во пократкиот од форматите %e и %f
%g	реален број во пократкиот од форматите %E и %f
%u	цел број без предзнак
%o	октален цел број без предзнак
%x	хексадецимален цел број без предзнак (мали букви)
%X	хексадецимален цел број без предзнак (мали букви)
%p	прикажува покажувач
%n	бројот на испишани знаци се доделува на аргументот
%%	испишување на знакот %

Примена на функцијата printf

Пример 5

```
#include <stdio.h>
int main() {
   printf(" e zbor dolg %d bukvi.\n", printf("Makedonija"));
   return 0;
}
```

Задача 1

Да се напише програма која ќе ја пресметува вредноста на математичкиот израз: $x = \frac{3}{2} + (5 - \frac{46*5}{12})$

Решение

```c
#include <stdio.h>
int main() {
    float x = 3.0 / 2 + (5 - 46 * 5 / 12.0);
    printf("x = %.2f\n", x);
    return 0;
}
```

Задача 2

Да се напише програма која за зададена вредност на x (при декларација на променливата) ќе го пресмета и отпечати на екран x^2.

Решение

```
#include <stdio.h>
int main() {
    int x = 7;
    printf("Number %d squared is: %d\n", x, x * x);
    return 0;
}
```

Задача 3

Да се напише програма која за дадени страни на еден триаголник ќе ги отпечати на екран периметарот и квадратот од плоштината (нека се работи со a=5, b=7.5, c=10.2).

Решение

```
#include <stdio.h>
int main() {
    float a = 5;
    float b = 7.5;
    float c = 10.2;
    float L = a + b + c;
    float s = L / 2;
    float P = s * (s - a) * (s - b) * (s - c);
    printf("Perimeter is: %.2f\n", L);
    printf("Area is: %.2f\n", P);
    return 0;
}
```

Задача 4

Да се напише програма за пресметување на аритметичката средина на броевите 3, 5 и 12.

Решение

```c
#include <stdio.h>
int main() {
    int a = 3, b = 5, c = 12;
    float am = (a + b + c) / 3.0;
    printf("Arithmetic mean is: %.2f\n", am);
    return 0;
}
```

Задача 5

Да се напише програма која ќе ги отпечати на екран остатоците при делењето на бројот 19 со 2, 3, 5 и 8.

Решение

```
#include <stdio.h>
int main() {
    int a = 19;
    printf("Remainder from division of %d with 2: %d\n", a, a %
           % 2);
    printf("Remainder from division of %d with 3: %d\n", a, a % 3);
    printf("Remainder from division of %d with 5: %d\n", a, a % 5);
    printf("Remainder from division of %d with 8: %d\n", a, a % 8);
    return 0;
}
```

Содржина

Елементи на околините за развој

Околината за развој е составена од повеќе програми, кои го олеснуваат целокупниот развој на една програма

- текст уредувач (text editor)
- преведувач (компајлер - compiler)
- дебагер (debugger)
- интеграција на библиотеки со функции
- поврзувач (linker)

Текст уредувач (text editor)

- Програма која овозможува внесување и уредување на текстот на изворната програма
- Овозможува зачувување на програми и вчитување на веќе напишани програми за нивно повторно уредување
- Означување на клучните зборови и команди во изворната програма (syntax highlighting)

Преведувач (compiler)

- Ја преобразува (преведува) изворната програма од јазикот за програмирање во кој е напишана во јазик разбирлив за компјутерот
- Се разликуваат два вида преведувачи: **интерпретери** и **компајлери**
- Интерпретер е преведувач кој ја *обработува одделно секоја команда*, ја проверува за грешки и ја извршува, по што поминува на следната команда итн.
- Компајлер е преведувач кој ја *обработува целата програма*, ја проверува за грешки и ја преведува, по што се добива извршната програма.
  - Така добиената извршна програма може да се извршува

Дебагер (debugger)

- Компајлерите и интерпретерите ги откриваат грешките (синтаксички) во програмата поради не правилно користење на јазикот за програмирање
- Друг вид на грешки се логичките грешки
  - Програмата не го прави тоа за кое што е наменета
  - Се откриваат многу тешко
- Дебагер е програма која помага при барање на логичките грешки
  - Овозможува следење на извршувањето на програмата чекор по чекор

Интеграција на библиотеки со функции

- Интегрирање и користење на претходно создадени и проверени модули (потпрограми), уште наречени и функции
- Ваквиот начин на организација на програмите има голем број на предности
- Повторно искористување на готови функционалности
- Пример библиотеки
  - За управување со стандардниот влез и излез
  - За стандардни математички операции

Поврзувач (linker)

- Понекогаш програмата е премногу голема за да се напише во една датотека
  - различните делови може да се пишуваат од различни програмери.
  - некои делови од дадена програма можат да бидат искористени и во друга програма
  - Одделно компајлираните делови е неопходно да бидат обединети во една цела извршна програма со помош на **поврзувачот**
  - Друга улога на поврзувачот е да ги „сврзе“ со програмата потребните библиотеки со стандардните функции

Околини за развој

(ntegrated Development Environment - IDE

- Сите овие елементи на околината за развој се обединуваат (интегрираат) во т.н. интегрирани околини за развој
- Пример за IDE е околината која ќе се користи на овој курс, Code::Blocks

Содржина

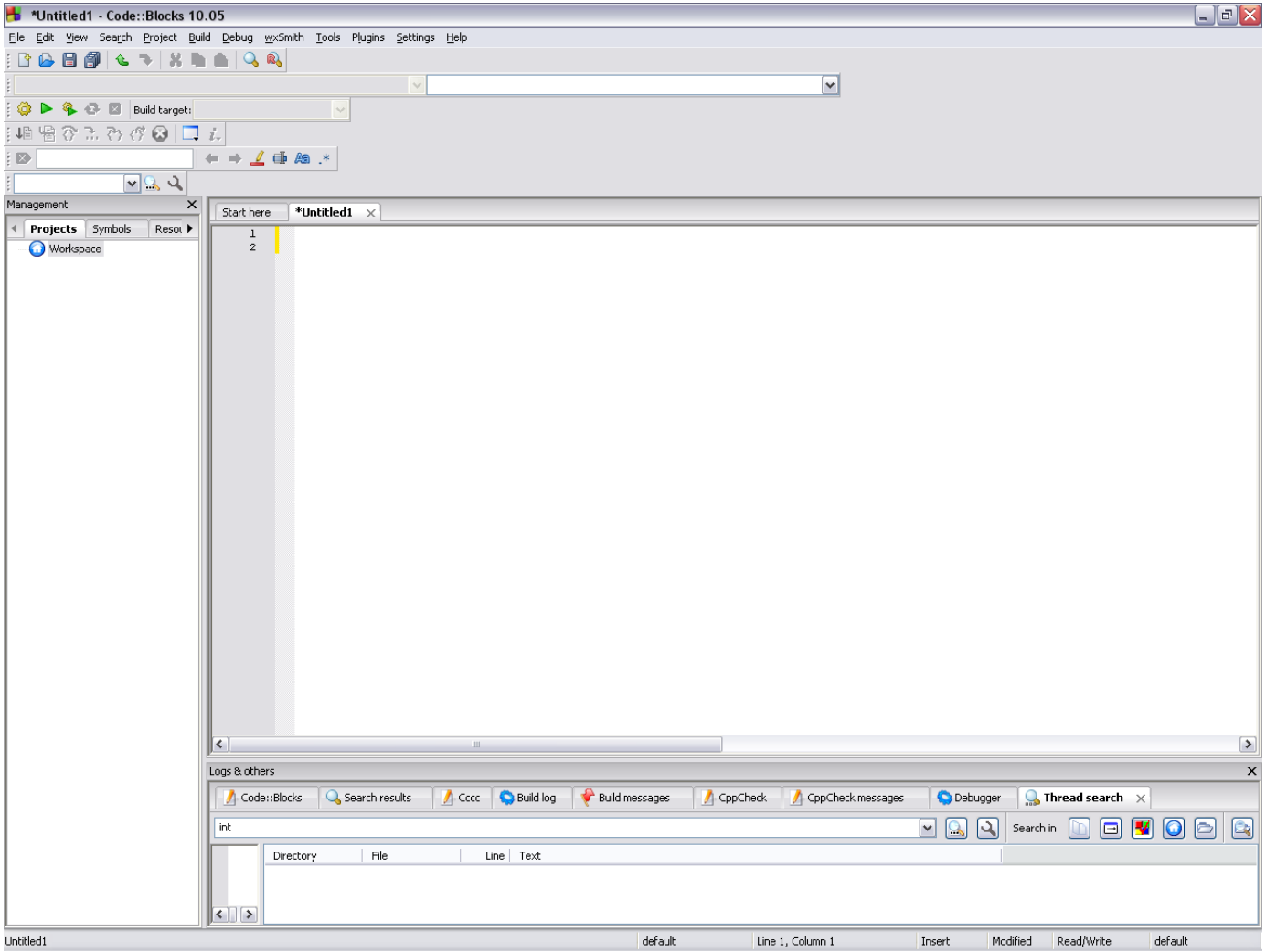
Code::Blocks - инсталација

- Како да го најдеме и инсталираме Code::Blocks
- Code::Blocks е **слободен софтвер** и може да се најде на http://www.codeblocks.org/downloads
- Во централниот дел на страната има три линка: **Download the binary release**, Download the source code и Retrieve source code from SVN
- За наједноставна инсталација се препорачува да се избере првиот линк - **Download the binary release**,

Code::Blocks – инсталација (2)

- За почетниците се препорачува да ја симнат верзијата што во нејзе вклучува **MinGW** setup
  - моментално тоа е линкот **codeblocks-10.05mingw-setup.exe** кој е наменет за корисниците на сите **Windows** оперативни системи
  - Со клик на изворот Sourceforge.net се отвора нова страницата која по истекот на 5 секунди сама ќе ви понуди опција да ја зачувате датотеката на од вас избрана локација
  - По зачувувањето на датотеката следете ги инструкциите за инсталирање

Code::Blocks – главен прозорец



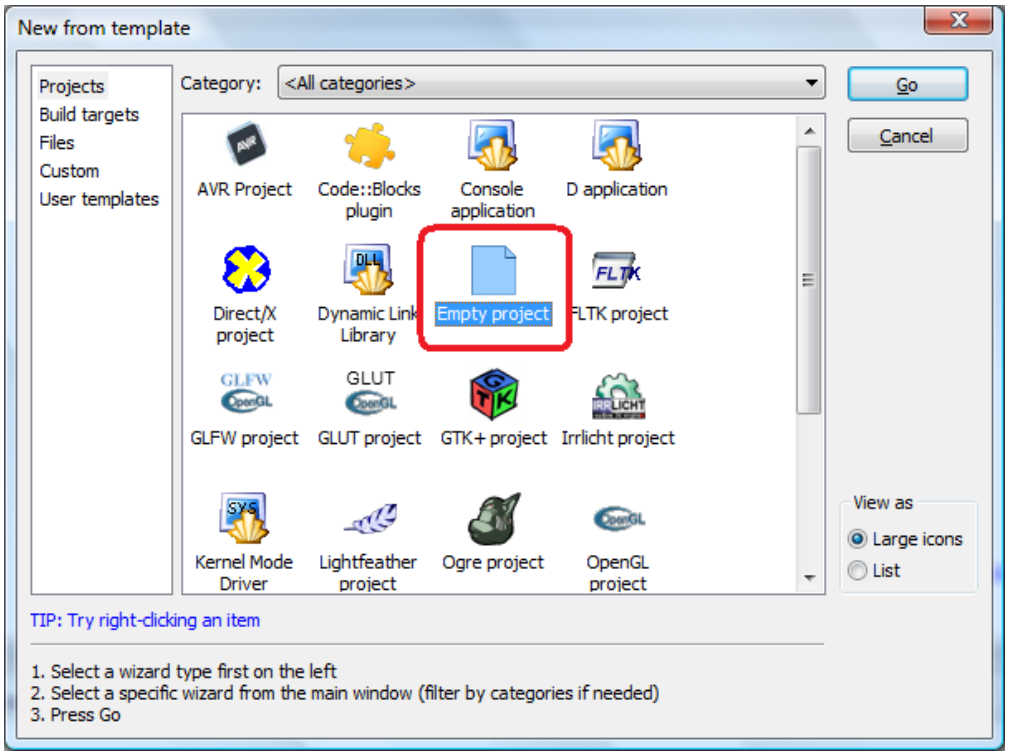
Елементи на главниот прозорец

- Лента со менија
  - лентата со менија се наоѓа во најгорниот дел на прозорецот, веднаш под неговиот насловот
  - Во неа се наоѓаат менијата File, Edit, View, Search, Project, Build, Debug, wxSmith, Tools, Plugins, Settings, Help
- Лента со алатки
  - лентите со алатки (копчиња за стартување на најчесто користените команди на околината) се наоѓаат непосредно под лентата со паѓачки менија
- Работна површина
  - Потпрозорец за уредувачот на текст
  - Прозорец за соопштенија.
  - Прозорец за организација на работата на програмата

Програмирање во C со Code::Blocks

Креирање проект

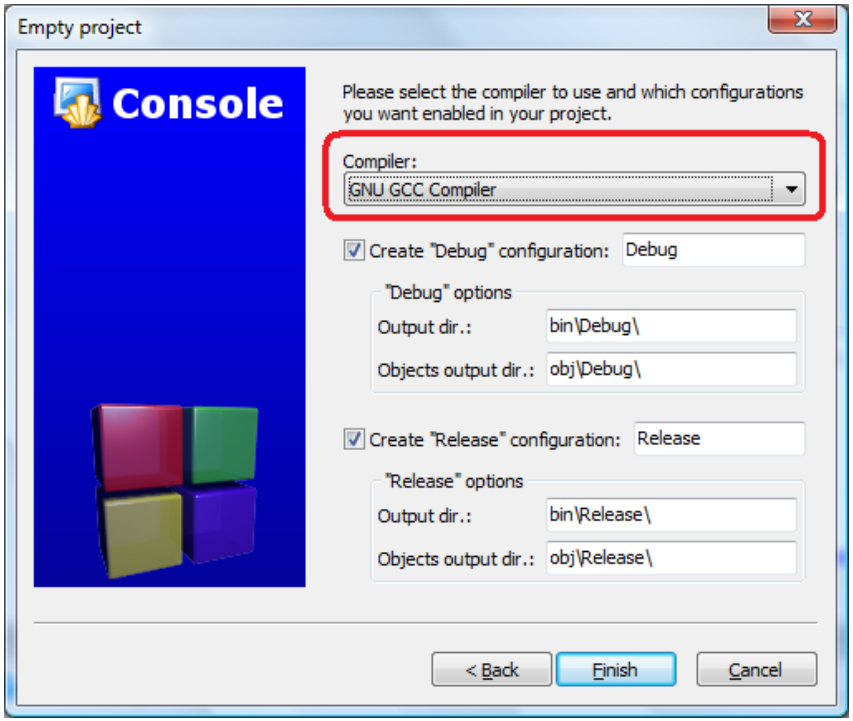
1. Стартувајте CodeBlocks
2. File -> New -> Project -> Empty Project -> Go



Програмирање во C со Code::Blocks

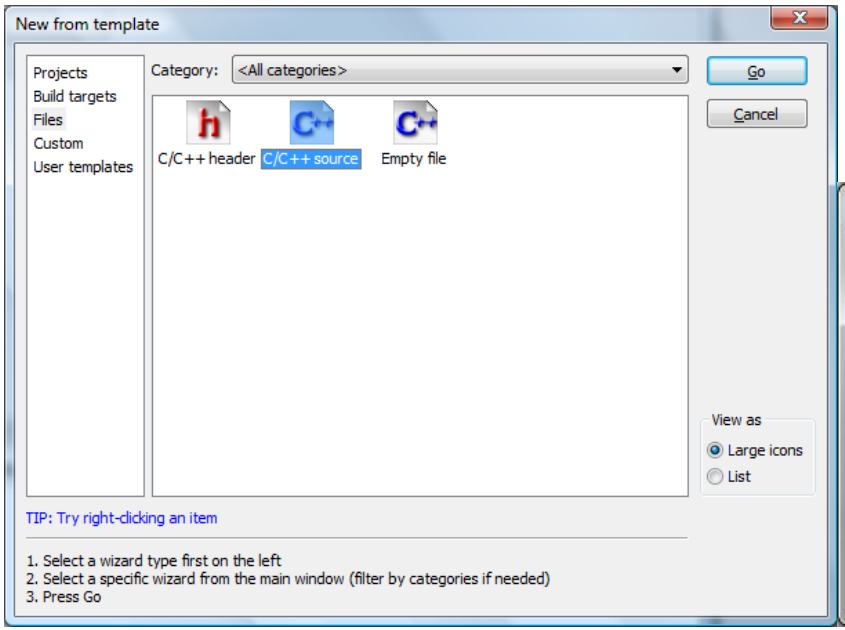
Креирање проект

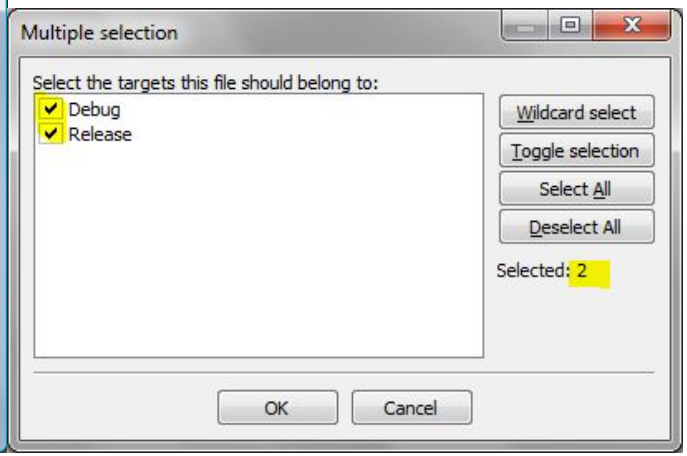
3. Одберете GNU GCC Compiler
4. Изберете ги следните 2 опции ако сакате да креирате “debug” и “release” configuration



Додавање на изворна датотека

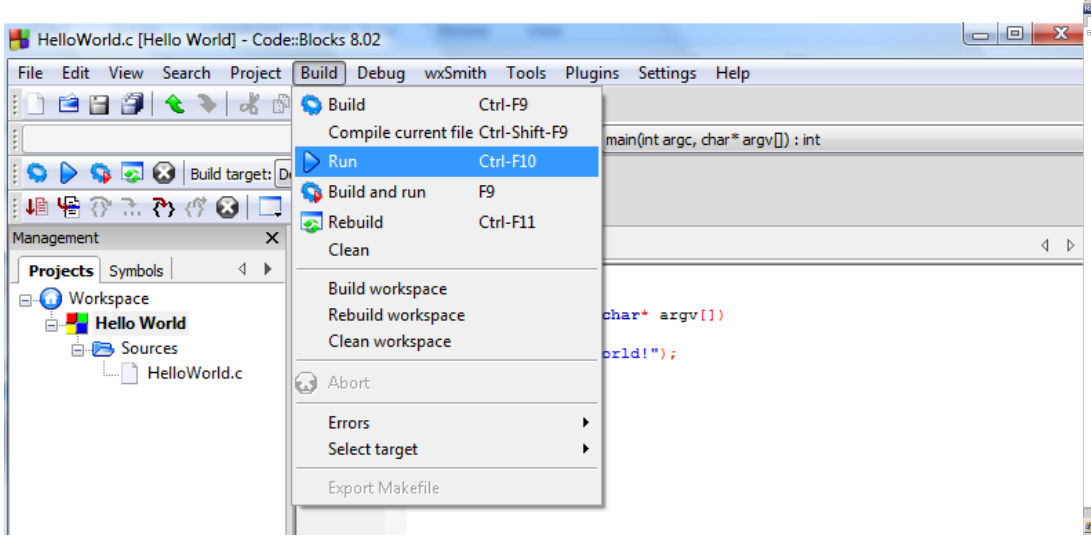
5. Додадете изворна датотека во проектот: File -> New -> File -> C/C++ Source
6. Одберете C како програмски јазик
7. Внесете го името на датотеката со полната патека и не заборавајте да го вклучите "Add file to active project"

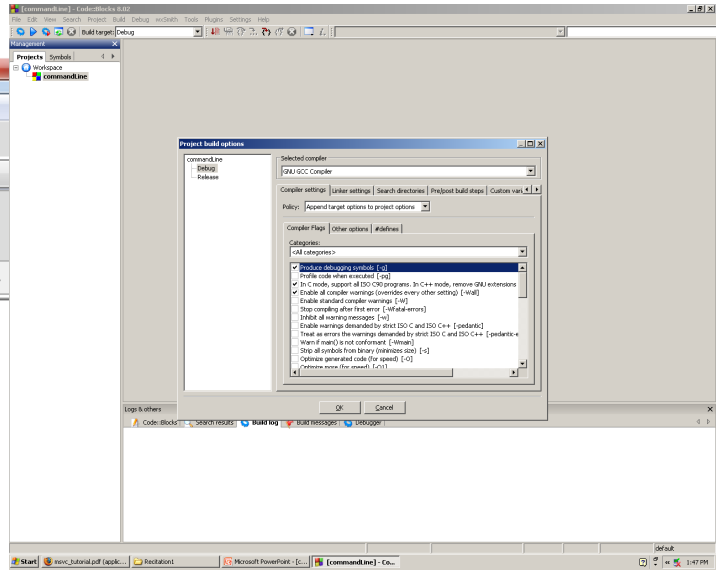




Програмирање

- За секој проект може да се постават следните опции "Project Build Options.. Compiler Flags"
- За изградба на проектот (build) притиснете Ctrl + F9
- За извршување на проектот притиснете Ctrl + F10





Задачи за дома

- Во продолжение се наведени неколку задачи кои би требало да се обидете да ги изработите дома
- Со нивна изработка ќе бидете подготвени за успешна работа на претстојните лабораториски вежби

Задача 1

Обидете се да креирате нов проект со една .c датотека и во неа внесете го текстот на следнава програма:

```c
#include <stdio.h>

int main() {
    printf("Zdravo, kako si?\n");
    return 0;
}
```

Задача 1

- Извршете ја програмата
  - Што добивате како резултат?
- Доколку сте направиле грешка при пишувањето на текстот поправете и извршете уште еднаш.
- Направете намерно некоја грешка во текстот. Извршете повторно!
  - Што се случува сега?

Задача 2

Во текстот на програмата додадете го означениот ред:

```
#include <stdio.h>
int main() {
    printf("Zdravo, kako si?\n");
    printf("Neshto ne ti se pravi muabet?\n");
    return 0;
}
```

Koj е резултатот од извршувањето сега?

Материјали и прашања

Предавања, аудиториски вежби, соопштенија
courses.finki.ukim.mk

Изворен код на сите примери и задачи
https://github.com/tdelev/SP/tree/master/latex/src

Прашања и дискусија
forum.finki.ukim.mk