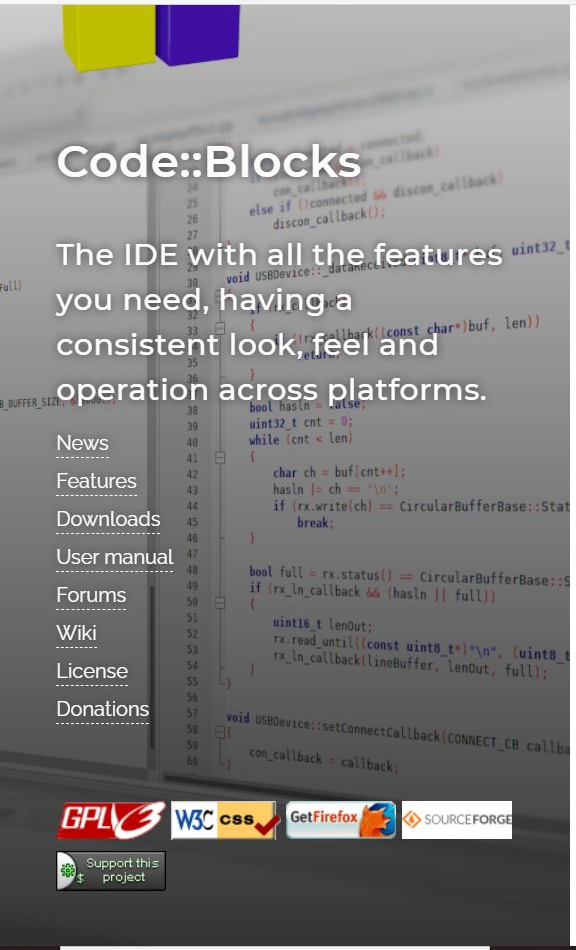
C++

http://www.codeblocks.org



Code::Blocks

The free C/C++ and Fortran IDE.

Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided by plugins!

If you 're new here, you can read the **user manual** or visit the **Wiki** for documentation. And don't forget to visit and join our **forums** to find help or general discussion about Code::Blocks.

We hope you enjoy using Code::Blocks!

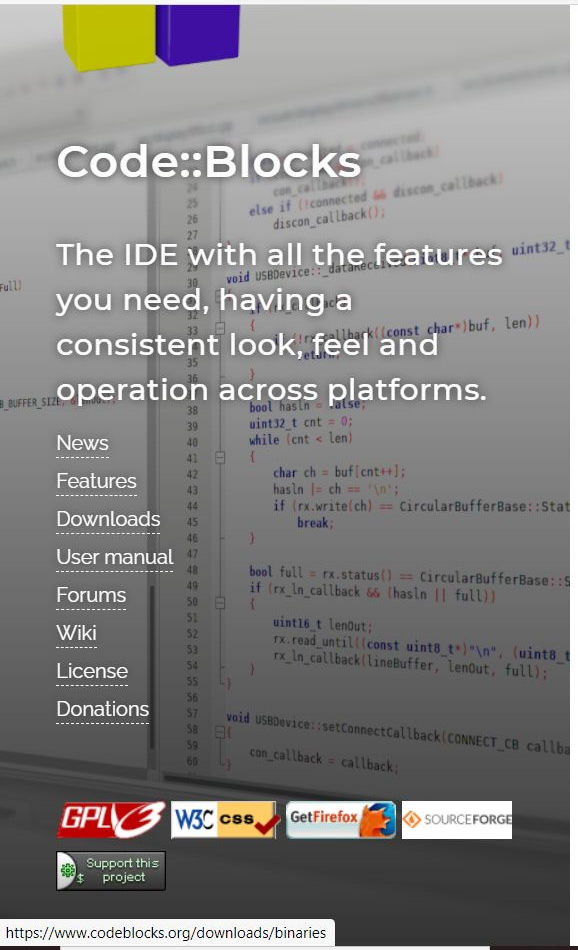
The Code::Blocks Team

Latest news

Migration successful

We are very happy to announce that the process of migrating to the new infrastructure has completed successfully!

Read more



Downloads

There are different ways to download and install Code::Blocks on your computer:

- ## Download the binary release

  This is the easy way for installing Code::Blocks. Download the setup file, run it on your computer and Code::Blocks will be installed, ready for you to work with it. Can't get any easier than that!

  - ### Download a nightly build

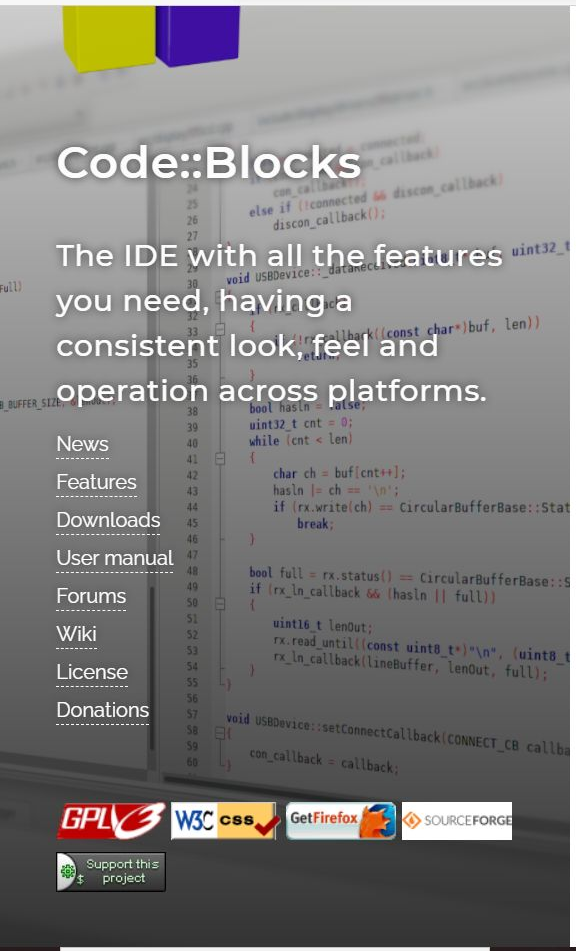
    There are also more recent so-called nightly builds available in the **forums**. Please note that we consider nightly builds to be stable, usually, unless stated otherwise.

  - Other distributions usually follow provided by the community (big "Thank you!" for that!). If you want to provide some, make sure to announce in the forums such that we can put it on the official C::B homepage.

- ## Download the source code

  If you feel comfortable building applications from source, then this is the recommend way to download Code::Blocks. Downloading the source code and building it yourself puts you in great control and also makes it easier for you to update to newer versions or, even better, create patches for bugs you may find and contributing them back to the community so everyone benefits.

- ## Retrieve source code from SVN



NOTE: There are also more recent nightly builds available in the forums or (for Ubuntu users) in the Ubuntu PPA repository. Please note that we consider nightly builds to be stable, usually.

NOTE: We have a Changelog for 20.03, that gives you an overview over the enhancements and fixes we have put in the new release.

NOTE: The default builds are 64 bit (starting with release 20.03). We also provide 32bit builds for convenience.

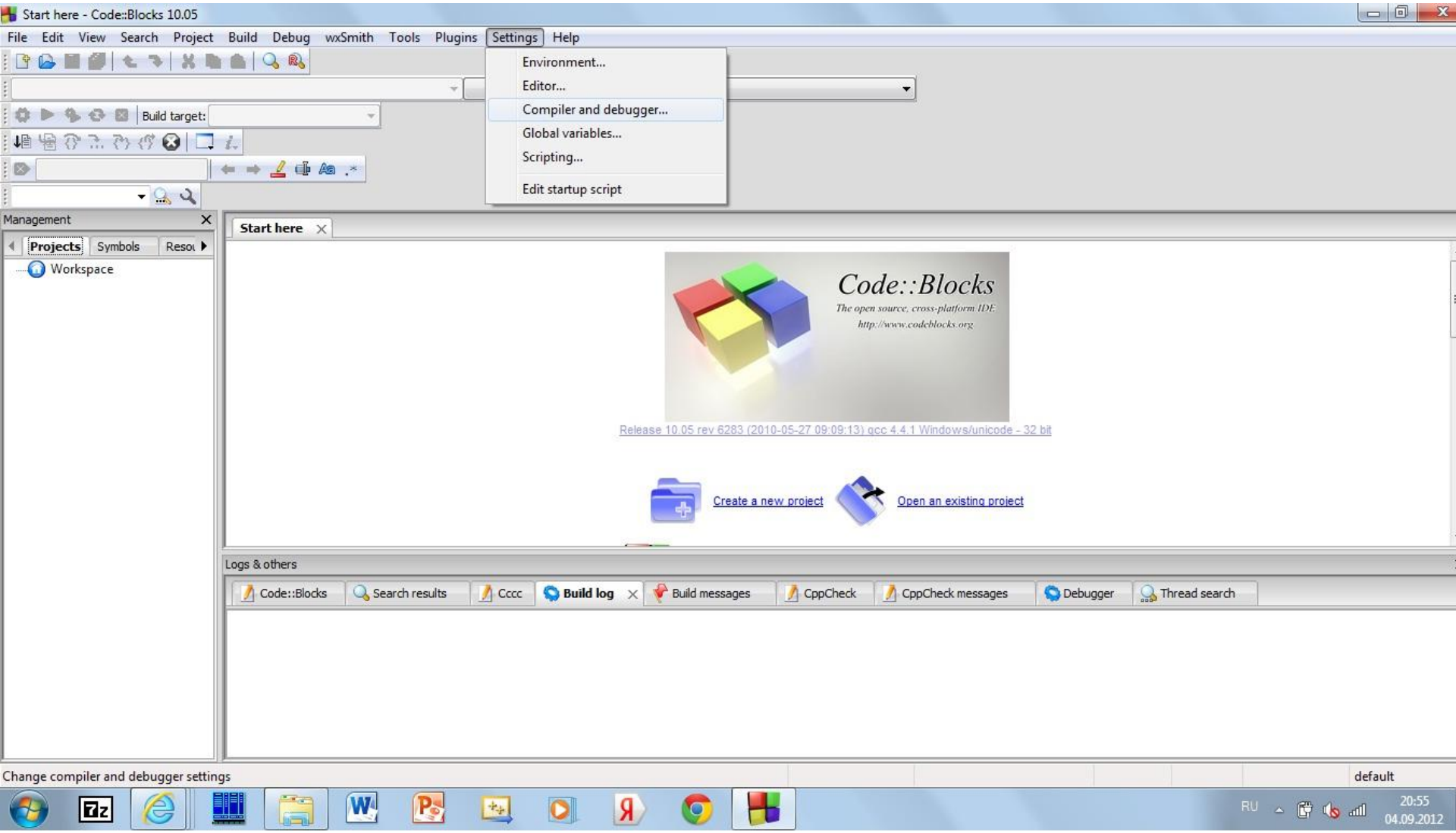
Microsoft Windows

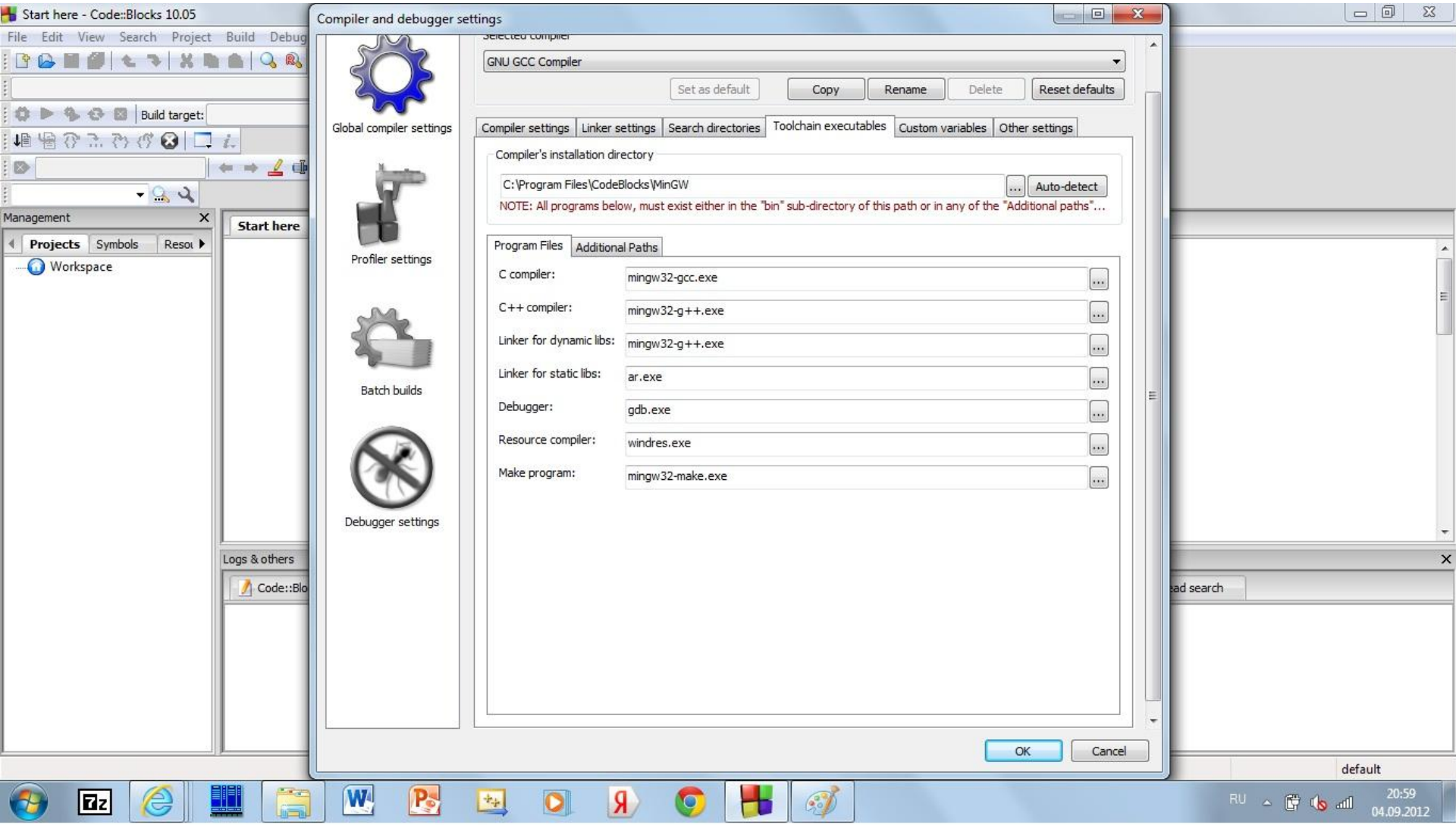
File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

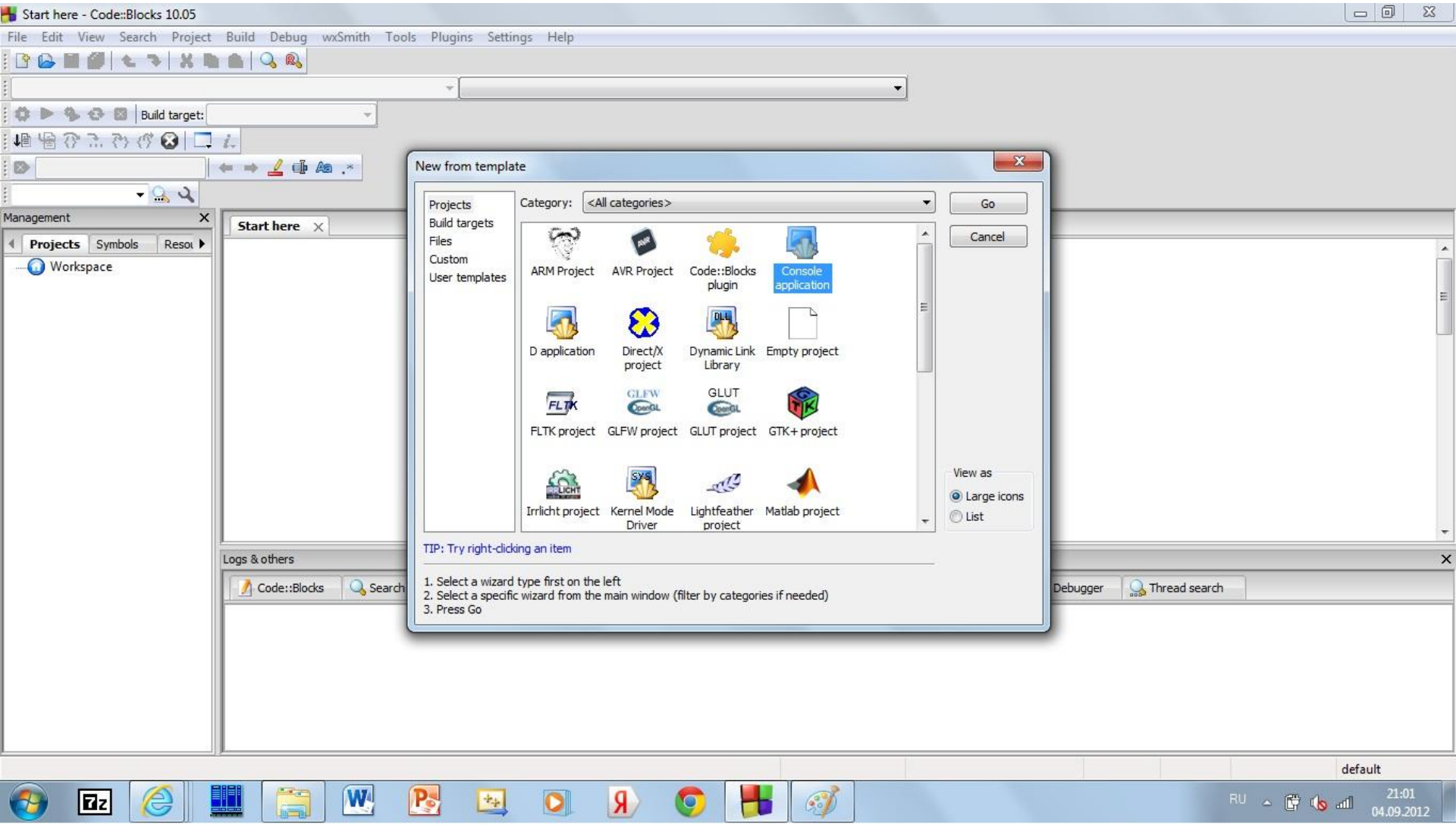
NOTE: The codeblocks-20.03-setup.exe file includes Code::Blocks with all plugins. The codeblocks-20.03-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).

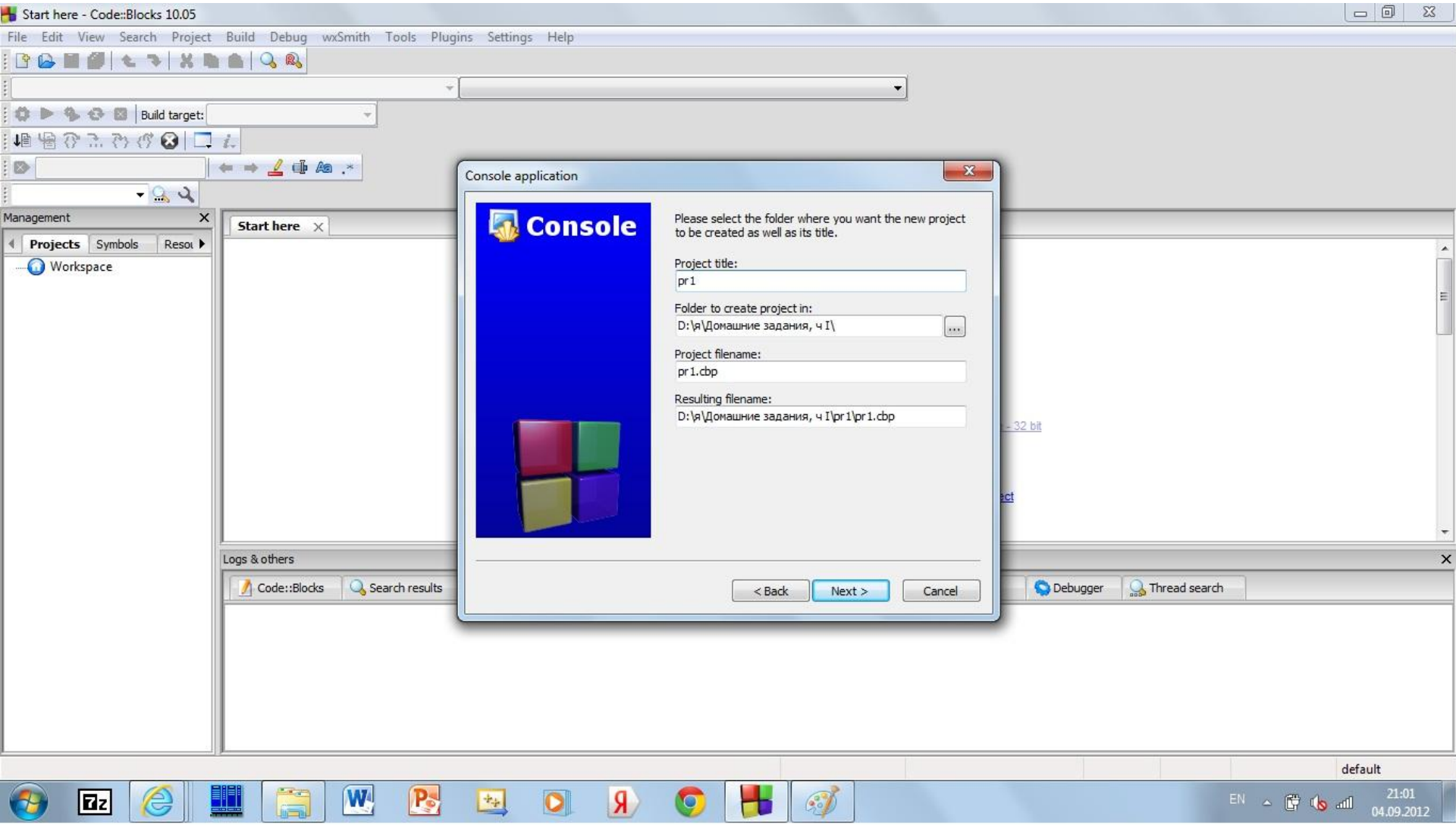
NOTE: The codeblocks-20.03mingw-setup.exe file includes additionally the GCC/G++/GFortran compiler and GDB debugger from MinGW-W64 project (version 8.1.0, 32/64 bit, SEH).

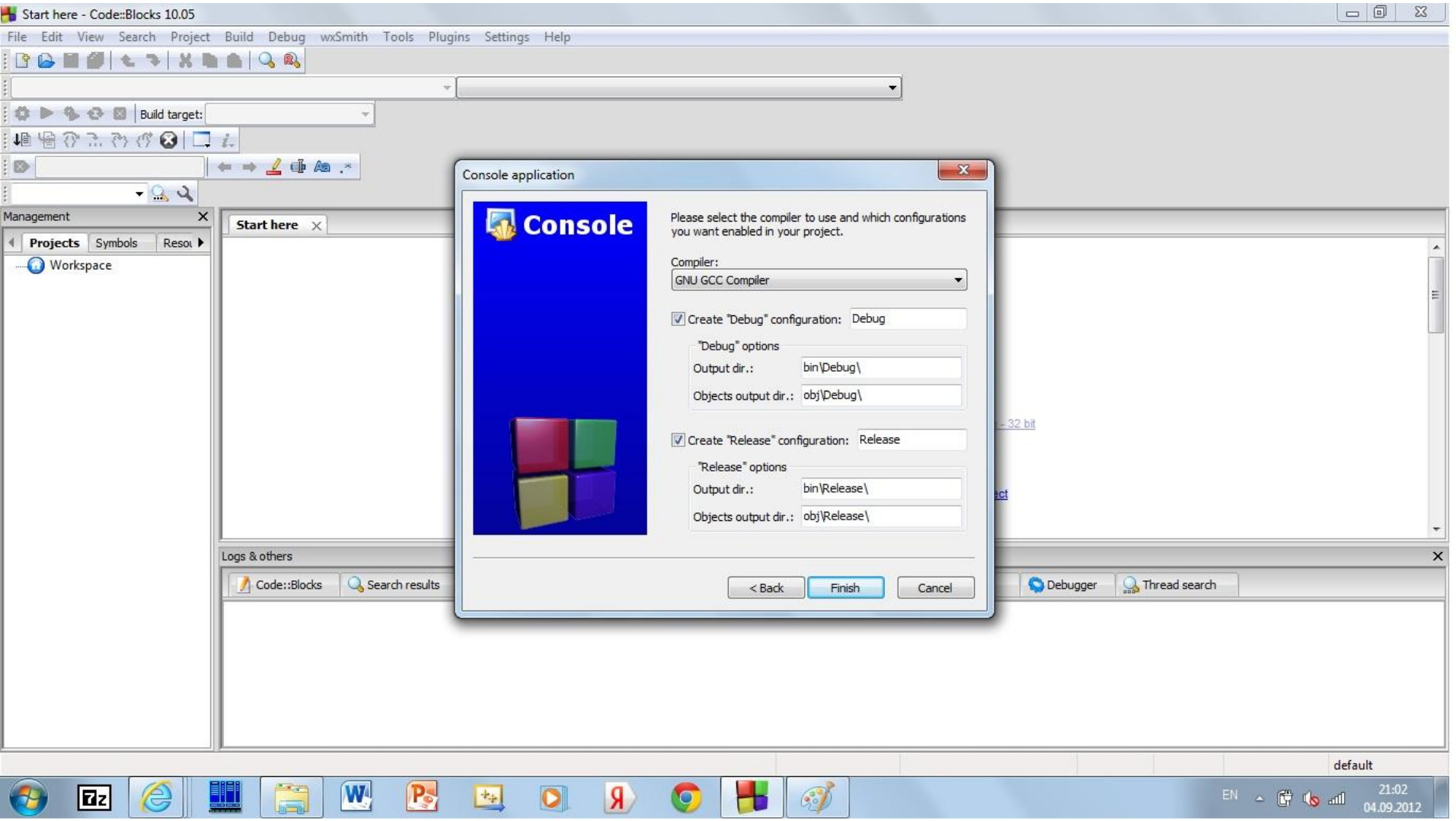
Создание и сохранение проекта

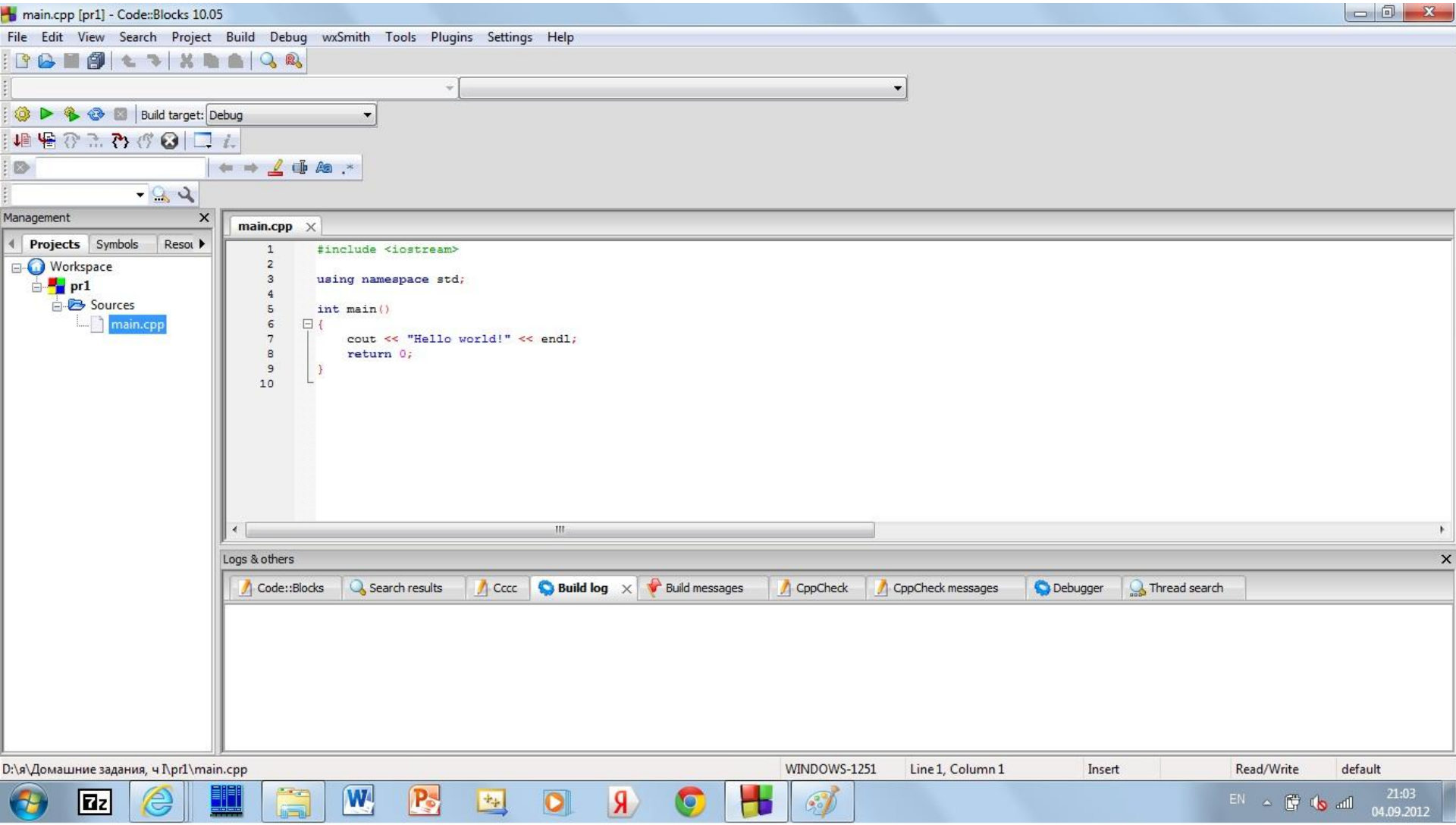

Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Environment...
Editor...
Compiler and debugger...
Global variables...
Scripting...
Edit startup script
Build target:
Management
Projects Symbols Resou
Workspace
Start here
Code::Blocks
The open source, cross-platform IDE
http://www.codeblocks.org
Release 10.05 rev 6283 (2010-05-27 09:09:13) gcc 4.4.1 Windows/unicode - 32 bit
Create a new project
Open an existing project
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck messages
Debugger
Thread search
Change compiler and debugger settings
default
RU
20:55
04.09.2012

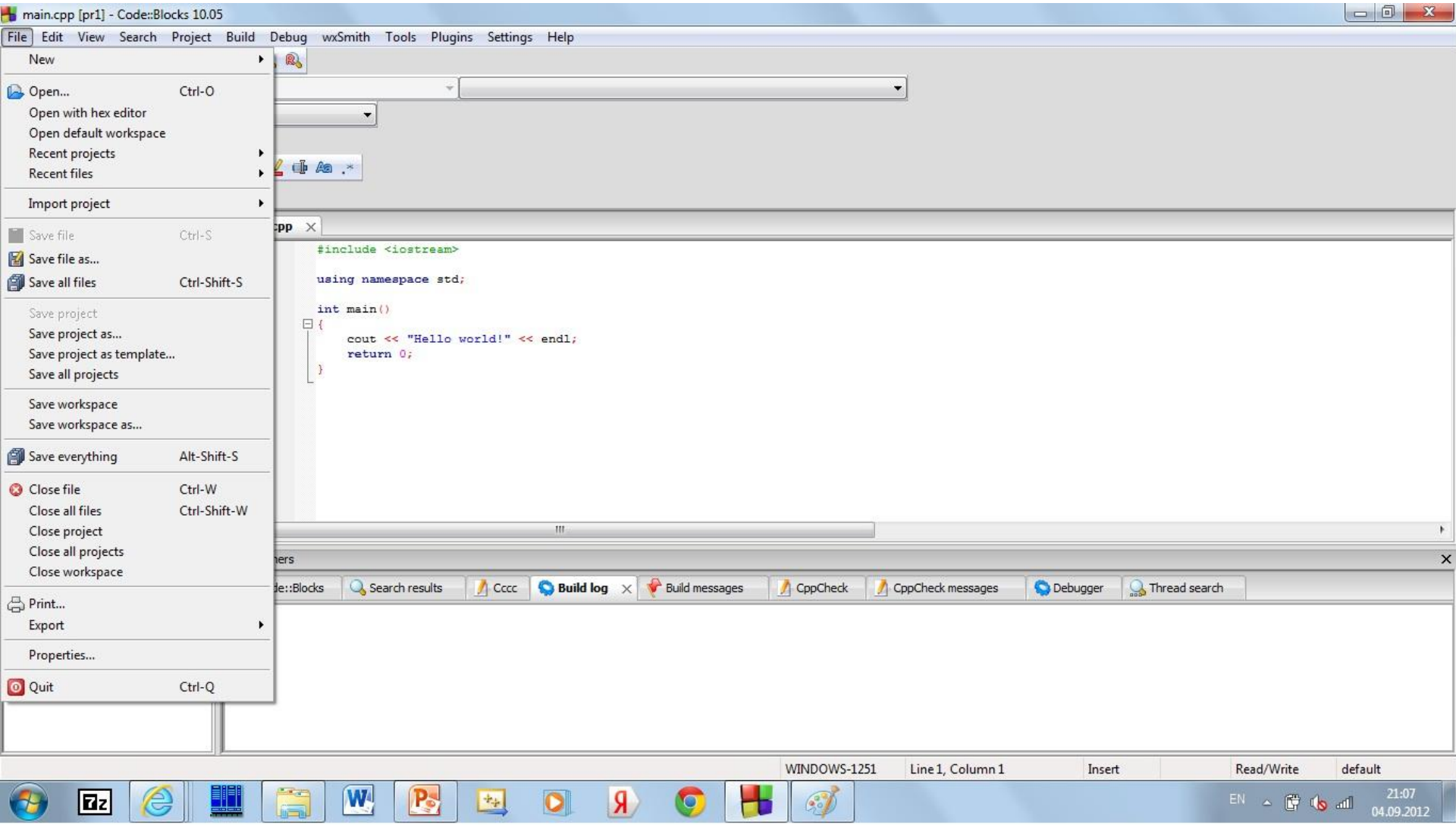

Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug
Build target:
Management
Projects Symbols
Workspace
Start here
Logs & others
Compiler and debugger settings
Global compiler settings
Profiler settings
Batch builds
Debugger settings
GNU GCC Compiler
Set as default
Copy
Rename
Delete
Reset defaults
Compiler settings
Linker settings
Search directories
Toolchain executables
Custom variables
Other settings
Compiler's installation directory
C:\Program Files\CodeBlocks\MinGW
Auto-detect
NOTE: All programs below, must exist either in the "bin" sub-directory of this path or in any of the "Additional paths"...
Program Files
Additional Paths
C compiler:
mingw32-gcc.exe
C++ compiler:
mingw32-g++.exe
Linker for dynamic libs:
mingw32-g++.exe
Linker for static libs:
ar.exe
Debugger:
gdb.exe
Resource compiler:
windres.exe
Make program:
mingw32-make.exe
OK
Cancel
default
RU
20:59
04.09.2012


Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Projects Symbols Resou
Workspace
Start here
New from template
Projects
Build targets
Files
Custom
User templates
Category: <All categories>
Go
Cancel
ARM Project
AVR Project
Code::Blocks plugin
Console application
D application
Direct/X project
Dynamic Link Library
Empty project
FLTK project
GLFW project
GLUT project
GTK+ project
Irrlicht project
Kernel Mode Driver
Lightfeather project
Matlab project
View as
Large icons
List
TIP: Try right-clicking an item
1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go
Logs & others
Code::Blocks
Search
Debugger
Thread search
default
RU
21:01
04.09.2012


Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Projects Symbols
Workspace
Start here
Console application
Console
Please select the folder where you want the new project to be created as well as its title.
Project title:
pr1
Folder to create project in:
D:\я\Домашние задания, ч I\
Project filename:
pr1.cbp
Resulting filename:
D:\я\Домашние задания, ч I\pr1\pr1.cbp
< Back
Next >
Cancel
Logs & others
Code::Blocks
Search results
Debugger
Thread search
default
EN
21:01
04.09.2012

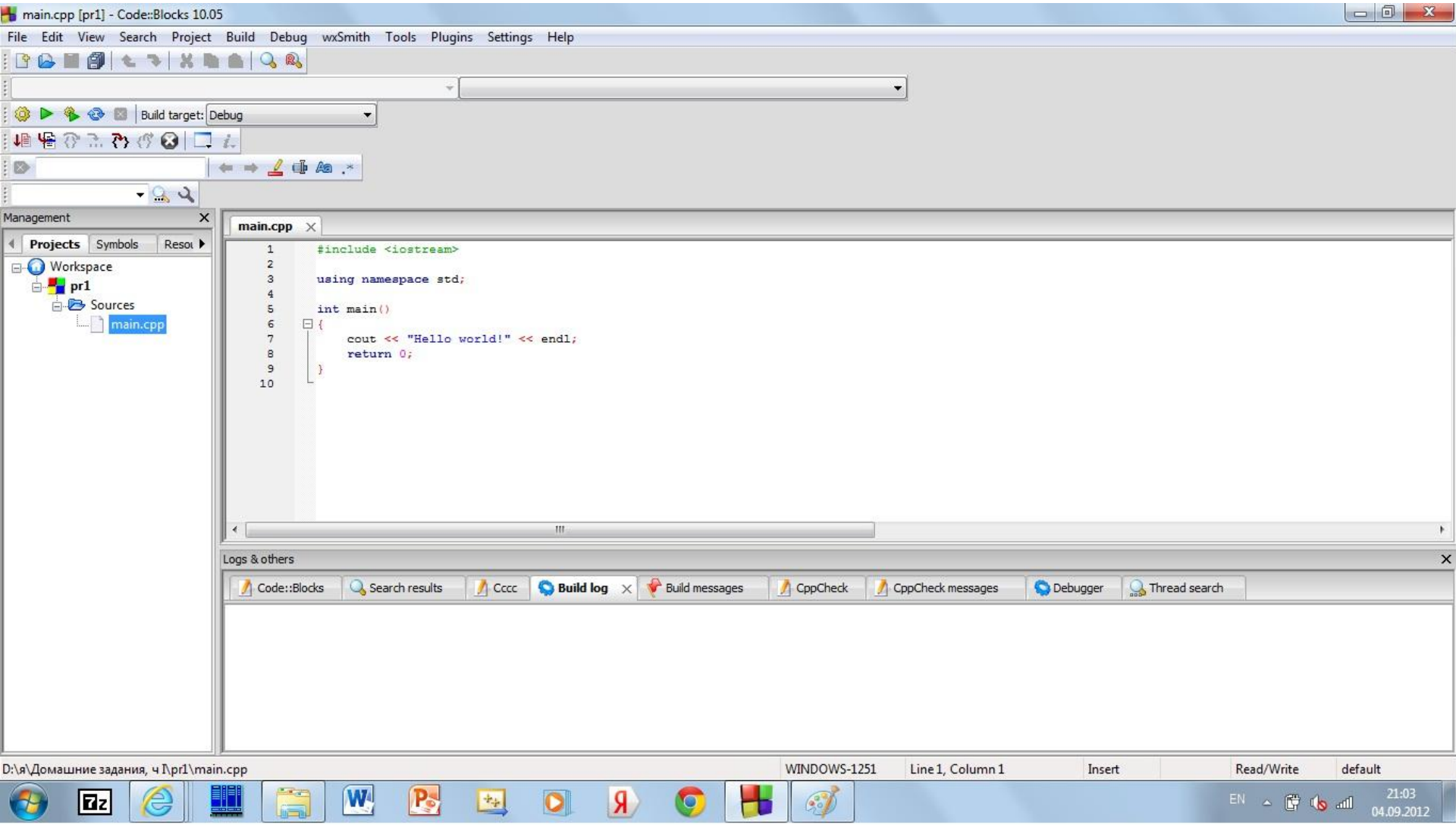

Start here - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target:
Management
Projects Symbols
Workspace
Start here
Logs & others
Code::Blocks
Search results
Debugger
Thread search
default
Console application
Console
Please select the compiler to use and which configurations you want enabled in your project.
Compiler:
GNU GCC Compiler
Create "Debug" configuration:
Debug
"Debug" options
Output dir.:
bin\Debug\
Objects output dir.:
obj\Debug\
Create "Release" configuration:
Release
"Release" options
Output dir.:
bin\Release\
Objects output dir.:
obj\Release\
< Back
Finish
Cancel
EN
21:02
04.09.2012


main.cpp [pr1] - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target: Debug
Management
Projects Symbols
Workspace
pr1
Sources
main.cpp
main.cpp
#include <iostream>
using namespace std;
int main()
{
cout << "Hello world!" << endl;
return 0;
}
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck CppCheck messages Debugger Thread search
D:\я\Домашние задания, ч I\pr1\main.cpp
WINDOWS-1251
Line 1, Column 1
Insert
Read/Write
default
EN
21:03
04.09.2012


main.cpp [pr1] - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
New
Open... Ctrl-O
Open with hex editor
Open default workspace
Recent projects
Recent files
Import project
Save file Ctrl-S
Save file as...
Save all files Ctrl-Shift-S
Save project
Save project as...
Save project as template...
Save all projects
Save workspace
Save workspace as...
Save everything Alt-Shift-S
Close file Ctrl-W
Close all files Ctrl-Shift-W
Close project
Close all projects
Close workspace
Print...
Export
Properties...
Quit Ctrl-Q
#include <iostream>
using namespace std;
int main()
{
cout << "Hello world!" << endl;
return 0;
}
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck messages
Debugger
Thread search
WINDOWS-1251
Line 1, Column 1
Insert
Read/Write
default
EN
21:07
04.09.2012

Язык С разработан Дэнисом Ритчи в начале 70-х годов под ОС UNIX

С++ разработан Бьерном Страуструпом в 1979 г.


main.cpp [pr1] - Code::Blocks 10.05
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target: Debug
Management
Projects Symbols
Workspace
pr1
Sources
main.cpp
main.cpp
#include <iostream>
using namespace std;
int main()
{
cout << "Hello world!" << endl;
return 0;
}
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck CppCheck messages Debugger Thread search
D:\я\Домашние задания, ч I\pr1\main.cpp
WINDOWS-1251
Line 1, Column 1
Insert
Read/Write
default
EN
21:03
04.09.2012

- **#include <iostream>** - организация ввода-вывода
- **using namespace std;** - поле имен
- **#include <bits/stdc++.h>** - стандартная библиотека. Здесь почти всё, что надо.

Переменная – именованная область памяти, в которой хранятся данные.

Максимальное количество символов - 31

Начинается с буквы. Верхние и нижние регистры различаются. Можно использовать латинские буквы, цифры и некоторые символы. Нельзя использовать ключевые слова в качестве переменных.

Данные – информация, которая хранится в ячейке памяти

Основные типы данных в C++. Типичные размеры значений и диапазоны представления

Тип	Размер в битах	Диапазон
`char`	8	-127-127 или 0-255
`wchar_t`	16	0-65 535
`int` (16-разрядная среда)	16	-32 768-32 767
`int` (32-разрядная среда)	32	-2 147 483 648-2 147 483 647
`float`	32	3,4E-38-3,4E+38
`double`	64	1,7E-308-1,7E+308
`bool`	-	`true` или `false`
`void`	-	Без значения

long long 64 бита (целое) $-2^{64} - 2^{63}$ -18446744073709551616
9223372036854775808

Объявление переменных

```
int i, j, k;

char ch, chr;

float f, balance;

double d;

 long long k;
```

Модификаторы типов

```
signed
unsigned
long
short
```

Операторы ввода/вывода

Ввод

```
cin >> a >>b;
int k, p; char c;
cin>>k>>c>>p;
```

5:6 k=5, c=‘:’, p=6

Вывод

```
cout << a << b;
cout << a << ‘ ‘ <<  b << endl;
 if(!cin.eof())
```

Задание ширины поля вывода

```
#include<iomanip>
cout << setw(10) << a;
cout << setfill('0') << setw(2) << h << ":" <<
setw(2) << m << endl;
```

Примеры использования различных типов выравнивания:

Пример кода	Вывод программы
cout << left << setw(7) << -123 << "*" << endl;	-123 *
cout << right << setw(7) << -123 << "*" << endl;	-123*
cout << internal << setw(7) << -123 << "*" << endl;	- 123*

Вывод действительных чисел

Для вывода действительных чисел в формате с фиксированной точкой используется манипулятор fixed, для вывода с плавающей точкой - манипулятор scientific.

По умолчанию числа выводятся с точностью в 6 знаков после точки

```
cout << scientific << setprecision(15) << x
<< endl;
```

Вывод/ввод по формату

printf(формат, переменная, переменная,…);

printf(“Сумма равна=%d\n”,s);

printf(“%d %d”,s1,s2);

scanf(“%d %d”, &s1, &s2);

форматы

%d – целое

%u – целое без знака

%p – указатель

%f – вещественное

%e – вещественное в экспоненциальной форме

%c – символ

%s – строка

%x – целое в шестнадцатеричной форме

Для указания длины поля %4d

Управляющие последовательности

\n – перевод на другую строку

\f – очистка экрана

\t – табуляция

\b – стирание символа перед курсором

Литералы - константы

- это фиксированные значения, которые не могут быть изменены программой. Они могут иметь любой базовый тип данных.

Символьные константы – например, ‘A’

Строковые константы – например, “abc”

Восьмеричные и шестнадцатеричные константы

Восьмеричные начинаются с 0

int a = 011; (9)

Шестнадцатеричные начинаются с 0x

int b = 0xFF; (255)

Инициализация переменных

- это присваивание им значений.

```
int a;
a = 10;

int a = 10;
```

Различные способы ввода данных.

Арифметические операторы

Оператор	Действие
+	Сложение
–	Вычитание, а также унарный минус
*	Умножение
/	Деление
%	Деление по модулю
--	Декремент
++	Инкремент

Инкремент и декремент

x = x + 1 x++ ++x (инкремент)

x = x - 1 x-- --x (декремент)

Префиксная форма ++x - сначала x увеличивается на 1, потом выполняются остальные операции.

Постфиксная форма x-- - сначала выполняются операции, а затем изменяется значение x.

Например

x = 10;

y = ++x; Результат x = 11

y = 11

x = 10;

y = x++ Результат x = 11

y = 10

Приоритеты

Приоритет	Операторы
Наивысший	++ --
	- (унарный минус)
	* / %
Низший	+ -

cmath

cmath — заголовочный файл стандартной библиотеки языка программирования C, разработанный для выполнения простых математических операций. Большинство функций привлекают использование чисел с плавающей точкой.

```
#include <cmath>
```

abs	Возвращает абсолютное значение числа
acos	Арккосинус
asin	Арксинус
atan	Арктангенс
ceil	округление до ближайшего большего целого числа

cos	Косинус
sin	Синус
tan	Тангенс
fabs	абсолютная величина (числа с плавающей точкой)
modf(*x*,*p*)	извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой

exp	вычисление экспоненты
log	натуральный логарифм
log10	логарифм по основанию 10
pow(*x*,*y*)	результат возведения *x* в степень *y*, x^y
sqrt sqrtl	квадратный корень

floor	округление до ближайшего меньшего целого числа
fmod	вычисление остатка от деления нацело для чисел с плавающей точкой
frexp	разбивает число с плавающей точкой на мантиссу и показатель степени.
ka=round(ak);	ak вещественное Округление до ближайшего целого
frexp	разбивает число с плавающей точкой на мантиссу и показатель

Выражения

Если в выражении присутствуют переменные и литералы разного типа, то компилятор приводит их к одному типу (с использованием типорасширения)

В C++ предусмотрена возможность установить для выражения заданный тип.

(Тип) выражение

(float) x/2

приведение типа – унарный оператор.

Логические переменные

Тип bool принимают два значения true(истина) или false(ложь)

Операторы отношений и логические операторы

Операторы отношений	Значение
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
Логические операторы	**Значение**
&&	И
||	ИЛИ
!	НЕ

p	q	p И q	p ИЛИ q	НЕ p
0	0	0	0	1
0	1	0	1	1
1	1	1	1	0
1	0	0	1	0

Приоритет	Операторы
Наивысший	!
	> >= < <=
	== !=
	&&
Низший	||

Инструкция if

```
if(выражение)
{
    последовательность инструкций
}
else
{
    последовательность инструкций
}
```

Выражение – некоторое действительное выражение, которое может быть интерпретировано как истинное или ложное. Числовое значение равное нулю, интерпретируется как ложное, не равное нулю - как истинное.

Пример

```
// Деление первого числа на второе.

#include <iostream>
using namespace std;

int main()
{
  int a, b;

  cout << "Введите два числа: ";
  cin >> a >> b;

  if(b) cout << a/b << '\n';
  else cout << "На нуль делить нельзя.\n";

  return 0;
}
```

Конструкция if – else - if

```
if(условие)
   инструкция;
else if(условие)
   инструкция;
else if(условие)
   инструкция;
.
.
.
else
   инструкция;
```

Пример

```
// Демонстрация использования "лестницы" if-else-if.
#include <iostream>
using namespace std;

int main()
{
  int x;

  for(x=0; x<6; x++) {
    if(x==1) cout << "x равен единице.\n";
    else if(x==2) cout << "x равен двум.\n";
    else if(x==3) cout << "x равен трем.\n";
    else if(x==4) cout << "x равен четырем.\n";
    else cout << "x не попадает в диапазон от 1 до 4.\n";
  }

  return 0;
}
```

Результаты выполнения этой программы таковы.

```
x не попадает в диапазон от 1 до 4.
x равен единице.
x равен двум.
x равен трем.
x равен четырем.
x не попадает в диапазон от 1 до 4.
```

Как видите, последняя `else`-инструкция выполняется только в том случае, если все предыдущие `if`-условия дали ложный результат.

Инструкция switch

Инструкция `switch` — это инструкция многонаправленного ветвления, которая позволяет выбрать одну из множества альтернатив.

```
switch(выражение) {
  case константа1:
    последовательность инструкций
    break;
case константа2:
    последовательность инструкций
    break;
  case константа3:
    последовательность инструкций
    break;
.
.
.
  default:
    последовательность инструкций
 }
```

Элемент *выражение* инструкции `switch` должен при вычислении давать целочисленное или символьное значение. (Выражения, имеющие, например, тип с плавающей точкой, не разрешены.) Очень часто в качестве управляющего `switch`-выражения используется одна переменная.

Инструкции `default`-ветви выполняются в том случае, если ни одна из `case`-констант не совпадет с результатом вычисления `switch`-выражения.

Итак, для применения `switch`-инструкции необходимо знать следующее.

- Инструкция `switch` отличается от инструкции `if` тем, что `switch`-выражение можно тестировать только с использованием условия равенства (т.е. на совпадение `switch`-выражения с заданными `case`-константами), в то время как условное `if`-выражение может быть любого типа.
- Никакие две `case`-константы в одной `switch`-инструкции не могут иметь идентичных значений.
- Инструкция `switch` обычно более эффективна, чем вложенные `if`-инструкции.

- Последовательность инструкций, связанная с каждой `case`-ветвью, *не* является блоком. Однако полная `switch`-инструкция определяет блок. Значимость этого факта станет очевидной после того, как вы больше узнаете о C++.

Цикл while

```
while(выражение)инструкция;
```

Цикл выполняется, если выражение=ИСТИНА

Выход из цикла – выражение = ЛОЖЬ

Цикл do – while

Цикл do-while — это единственный цикл, который всегда делает итерацию хотя бы один раз.

for , while – сначала проверяется условие, потом выполняются инструкции цикла;

do-while – сначала выполняются инструкции цикла, потом проверяется условие.

```
do {
    инструкции;
} while (выражение);
```

Цикл выполняется пока выражение = ИСТИНА

continue

средство “досрочного” выхода из текущей итерации цикла.

```
#include <iostream>
using namespace std;

int main()
{
  int x;

  for(x=0; x<=100; x++) {
    if(x%2) continue;
    cout << x << ' ';
  }

  return 0;
}
```

break

Немедленный выход из цикла

```
#include <iostream>
using namespace std;

int main()
{
  int t;

  // Цикл работает для значений t от 0 до 9, а не до 100!
  for(t=0; t<100; t++) {
    if(t==10) break;
    cout << t << ' ';
  }

  return 0;
}
```

Цикл for

```
for(инициализация; выражение; инкремент)
{
  последовательность инструкций
}
```

```
for(num=1; num < 100; num++) {
```

Инструкции

```
}
```

```
for(i=100; i >= -100; i = i-5) cout << i << ' ';
```

Вариации на тему цикла for

```
for(x=0, y=10; x<=10; ++x, --y)
  cout << x << ' ' << y << '\n';
```

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{    int i;
    for(i=0; !kbhit(); i++) cout << i << ' ';
      system("pause");
    return 0;  }
```

```
#include <iostream>
using namespace std;

int main()
{
  int x;

  for(x=0; x != 123; ) {
    cout << "Введите число: ";
    cin >> x;
  }

  return 0;
}
```

```
cout << "Введите номер позиции: ";
cin >> x;

for( ; x < limit; x++) cout << ' ';
```

Одномерные массивы

Одномерный массив — это список связанных переменных.

```
тип имя_массива[размер];
```

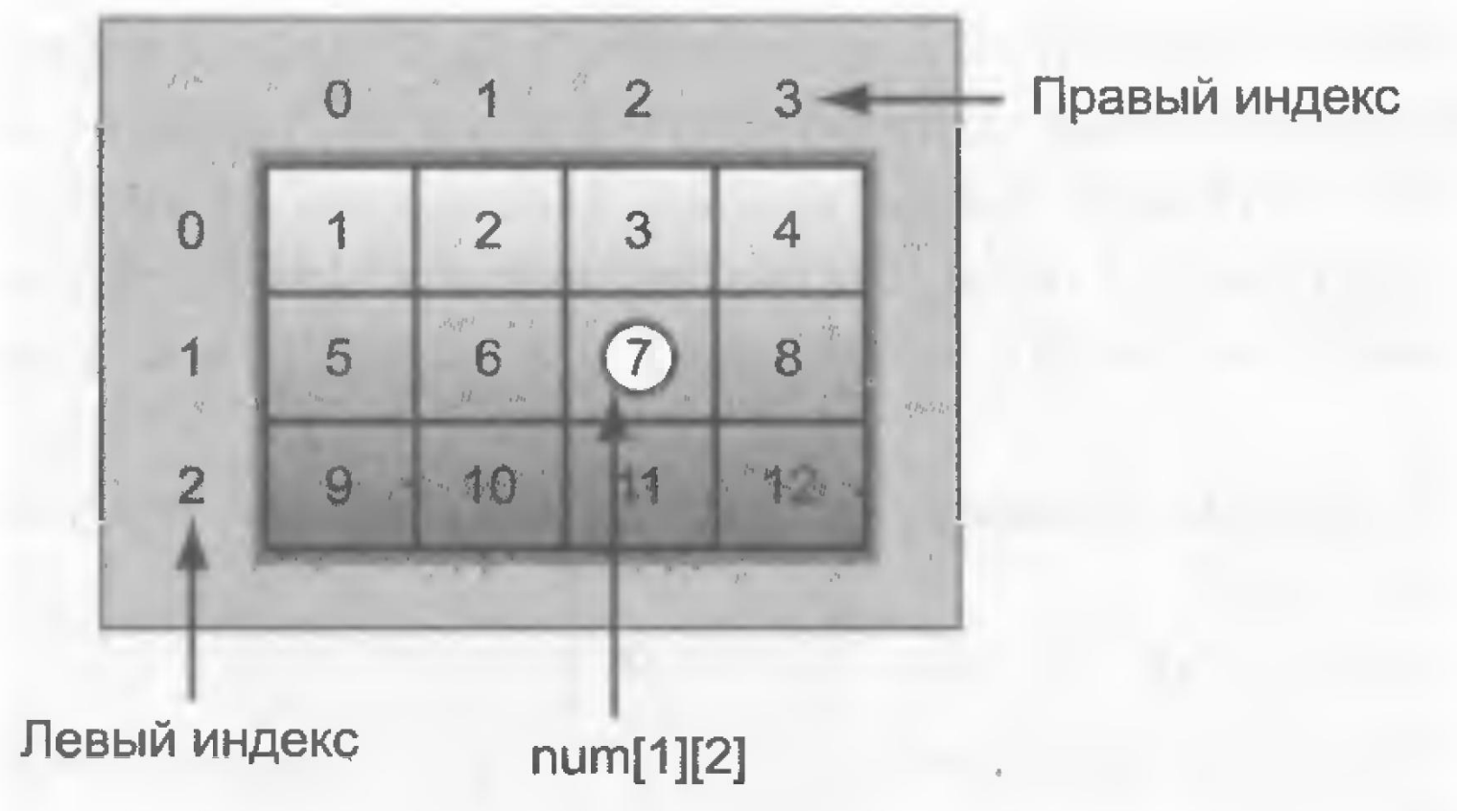
```
int sample[10];
```

```
#include <iostream>
using namespace std;

int main()
{
  int sample[10]; // Эта инструкция резервирует область
                  // памяти для 10 элементов типа int.
  int t;

  // Помещаем в массив значения.
  for(t=0; t<10; ++t) sample[t]=t;

  // Отображаем массив.
  for(t=0; t<10; ++t) cout << sample[t] << ' ';

  return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{ int n;
cin >> n;
int a[n];
int i;
for(i=0; i<n; i++)
a[i]=i;
for(i=0; i<n; i++)
cout << a[i] << ' ';
    return 0;
}
```

Двумерные массивы

```
int twod[10][20];
int t,i, num[3][4];
```



```
#include <iostream>

using namespace std;

int main()
{ int n, m;
cin >> n >> m;
int a[n][m];
int i, j;
for(i=0; i < n; i++)
 for(j=0; j < m; j++)
 cin >> a[i][j];
 for(i=0; i < n; i++)
 {for(j=0; j < m; j++)
 cout<< a[i][j] << ' ';
 cout << endl;}
return 0;
}
```

```
cin>>n;
int *a=new int[n];
```

Инструкция go to

Инструкция безусловного перехода.

Метка – это идентификатор, за которым стоит двоеточие.

```
x = 1;
   loop1:
      x++;
      if(x < 100) goto loop1;
```

Строки

Строка — это символьный массив, который завершается нулевым символом.

Строковый литерал

“Привет”

“” – нулевой литерал. Состоит только из нулевого символа – признака завершения строки

```
char str[11];
```

строки класса string

```
#include <string>
```

```
string st2, st3, s1, s2, s3;
операции присваивания:
st2 = st3; // копируем st3 в st2
s3 = s1 + s2;

s1 += s2;
```

Последний элемент ‘\0’

```
S[i]  = ‘a’;
S[i]= 97;
Int k = s[i];
Int k = s[i] - ’0’;
```

size	возвращает количество символов в строке	Int l=s.size(); l = s.length();
getline	Ввод с пробелами	getline(cin, str);
erase	Удаление заданного количества символов	s.erase(3, 5);
insert	Вставляет подстроку s2	s1.insert(3, s2);
replace	Замена	s2.replace(2, 4, s1); s2.replace(4, 2, s1, 0, 4);

find(s2)	Возвращает номер позиции первого (или первого, начиная с заданной позиции) вхождения или -1, если не найдено	pos = s1.find(s2); pos = s1.find(s2, 3);
rfind(s2)	Возвращает номер позиции последнего (или последнего, начиная с заданной позиции) вхождения или -1, если не найдено	pos = s1.rfind(s2); pos = s1.rfind(s2, 3);
substr	возвращает подстроку строки	S2= s.substr(start, [length]);

int res = stoi(s1);

Int k=count(s.begin(), s.end(), a); (в алгоритмах)

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main()
{
  string s, s1;
  getline(cin, s);
  s1=s;
  reverse(s.begin(), s.end());
  if(s==s1)
   cout<<"yes";
  else
   cout<<"no";
   return 0;
}
```

Векторы

```
#include <vector>
vector <int> a(n);
 vector <int> a;
```

vector <int> ivector = {<элемент [0]>, <[1]>, <[2]>};

```
vector <int> a(10)
for (auto now : a)
{      cout << now << "  ";}
```

Методы

- push_back a.push_back(temp);
- size() l=a. size()
- pop_back()
- clear() — удалить все элементы вектора
- empty() — проверить вектор на пустоту
- a.resize(10);
- a.insert (it,200);

```
a.erase(a.begin()+1);
///стираем 1 элемент вектора

   a.erase(a.begin()+2,a.begin()+6);
///стираем 3-6 элементы вектора,
7 элемент не стирается
```

Пары

```
#include <vector>
#include <algorithm>
#include <utility>

vector <pair <int, int>> a(n);
vector <pair <int, int>> a;
```

```
 for (int i = 0; i < n; i++) {
        int temp;
        cin >> temp;
        a[i] = {temp, i}; // создание пары значение - номер
    }
sort(a.begin(), a.end());
```

```
for (auto now : a) {
        cout << now.second << " ";
    }
    cout<<endl;
    for(int i=0; i<n; i++)
        cout<<a[i].first<<' ';
```

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
   vector <int> v1;
   vector <string> s1;
   string s;
   int l, a, i, k;
   cin>>l;
   for(i=0; i<l; i++)
   {
      cin>>s;
      s1.push_back(s);
   }
   cin>>s;
   k=count(s1.begin(), s1.end(),s);

   cout << k << endl;
   return 0;
}
```

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

int main()
{
   vector <int> v1;
   vector <string> s1;
   string s;
   int l, a, i, k;
   cin>>l;
   for(i=0; i<l; i++)
   {
      cin>>s;
      s1.push_back(s);
   }
   cin>>s;
vector<string>::iterator p;
      p = find(s1.begin(), s1.end(), s);
      k=p-s1.begin();
      cout << k <<' '<<s1[k]<< endl;
      s1.erase(p, p+1);

   for(auto x:s1)
      cout<<x<<' ';
   return 0;
}
```

Вектор векторов

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{vector < vector <int> > vec;
int i, j, n, a, m;
vector <int> v1;
cin>>n>>m;
//vector <int> v1(m);
for(i=0; i<n; i++)
{v1.clear();
   for(j=0; j<m; j++)
{
   cin>>a;
  //  v1[j]=a;
   v1.push_back(a);
}
vec.push_back(v1);
}
for(i=0; i<n; i++)
{for(j=0; j<m; j++)
cout<<vec[i][j]<<' ';
cout<<endl;}

   return 0;
}
```

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

int main()
{
//Удаление диапазона
   vector <int> v1, v2;
   int i, k1, k2;
   for(i=0; i<10; i++)
      v1.push_back(2*i);
   k1=1; k2=3;
   v1.erase(v1.begin()+k1, v1.begin()+k2);
   for(i=0; i<v1.size(); i++)
      cout<<v1[i]<<' ';
   cout<<endl;
//Удаление заданного значения
   for(i=0; i<10; i++)
      v2.push_back(2*i);
   std::vector<int>::iterator p;
   p = std::remove(v2.begin(), v2.end(), 10); //(или)
 //p = std::find(v2.begin(), v2.end(), 10);
   v2.erase(p, p+1);
   for(i=0; i<v2.size(); i++)
      cout<<v2[i]<<' ';
   cout<<endl;
//То же для пар
   vector <pair <int, int>> vp;
   pair <int, int> pr;

   for(i=0; i<10; i++)
   {
      pr.first=i;
      pr.second=2*i;
      vp.push_back(pr);
   }
/*
   vp.erase(vp.begin()+k1, vp.begin()+k2);
   for(i=0; i<vp.size(); i++)
      cout<<vp[i].first<<' '<<vp[i].second<<"  ";
   cout<<endl;
*/
   std::vector<pair <int, int>>::iterator p1;
   pr.first=3;
   pr.second=6;

   p1 = std::remove(vp.begin(), vp.end(), pr);
   vp.erase(p1, p1+1);
   for(i=0; i<vp.size(); i++)
      cout<<vp[i].first<<' '<<vp[i].second<<"  ";
   cout<<endl;
   return 0;
}
```

Remove удаляет заданный элемент, записывая остальные элементы по верх, длина вектора не изменяется.

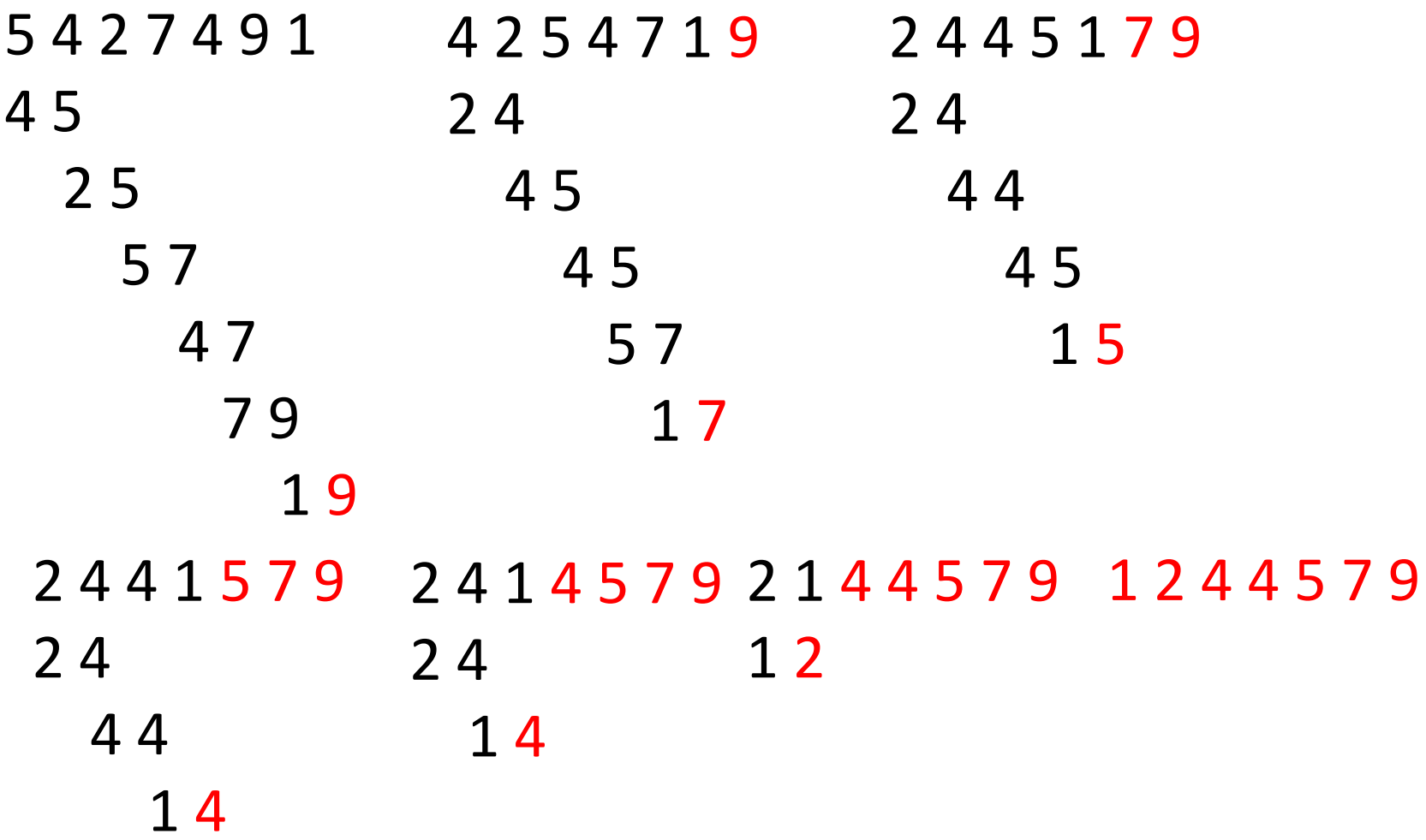
Int index=p-v2.begin();

- Турнир муниципальный 18-19

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{int n, i, k, p, f, m, r,a;
cin>>n;
   vector <int> s(n);
   vector <int> s1(0);
   for(i=0; i<n; i++)
      s[i]=i+1;
 p=n-1;
 k=0;
 f=1;
 while(f != n)
 {
    f*=2;
    k++;
 }
 m=n;
cout<<k<<endl;
  for(int g=0; g<k; g++)
  {
  m=m/2;
  for(int j=0; j<m; j++)
  {
     cin>>r;
     if(r==1) a=s[2*j];
     else
     a=s[2*j+1];
     s1.push_back(a);
  }
for (auto now : s1) { cout << now << " ";}
cout<<endl;

  s=s1;
 s1.clear();
  }
 cout<<s1[0];

   return 0;
}
```

Сортировки

Метод пузырька Время работы n^2



```
for(i=0; i<n; i++)
    for(j=0; j<n-i-1; j++)
    if(a[j] > a[j+1])
    {a1=a[j];
     a[j]=a[j+1];
     a[j+1]=a1;}
```

Быстрая сортировка

4 6 3 9 8 5 2 7 9 4 1 3 1 6 7

Выбор элемента 7

4 6 3 5 2 4 1 3 1 6 7 7 9 9 8

Выбор элементов 4 9

3 2 1 3 1 4 4 6 5 6 7 7 8 9 9

Выбор элементов 1 5

1 1 3 2 3 4 4 5 6 6 7 7 8 9 9

Выбор элементов 2

1 1 2 3 3 4 4 5 6 6 7 7 8 9 9

Время работы nlog(n)

```
#include <algorithm>
sort(a.begin(), a.end());
reverse(begin(a), end(a));
sort(a.rbegin(), a.rend());

sort(a.begin(), a.end(), cmp);
```

cmp - компаратор

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
bool fsort(int a, int b)
{
   if(a%2==0)
   return a < b;
   else
    return b < a;
}
int main()
{
   vector <int> v;
   int n, a;
   cin>>n;
   for(int i=0; i<n; i++)
   {
      cin>>a;
      v.push_back(a);
   }
   sort(v.begin(), v.end());
   for(int i=0; i<n; i++)
      cout<<v[i]<<' ';
   cout<<endl;

   sort(v.rbegin(), v.rend());
   for(int i=0; i<n; i++)
      cout<<v[i]<<' ';
   cout<<endl;

   sort(v.begin(), v.end(), fsort);
   for(int i=0; i<n; i++)
      cout<<v[i]<<' ';
   cout<<endl;

   return 0;
}
```

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
bool fsort(pair <int, int> a, pair<int, int> b)
{
   if(a.first != b.first)
   return a.first < b.first;
   if(a.first == b.first)
   return a.second<b.second;
}
int main()
{
   vector<pair<int, int>> v;
   pair <int, int> par;
   int n, a, b;
   cin>>n;
   for(int i=0; i<n; i++)
   {
      cin>>a>>b;
      par={a, b};
      v.push_back(par);
   }

   sort(v.begin(), v.end(), fsort);
   for(int i=0; i<n; i++)
      cout<<v[i].first<<' '<<v[i].second<<endl;
   cout<<endl;
   return 0;
}
```

Указатели

Указатель — это переменная, которая содержит адрес другой переменной.

Переменные-указатели (или *переменные типа указатель*) должны быть соответственно объявлены. Формат объявления переменной-указателя таков:

*тип *имя_переменной;*

```
int *p;
```

Базовый тип указателя определяет тип данных, на которые он будет ссылаться.

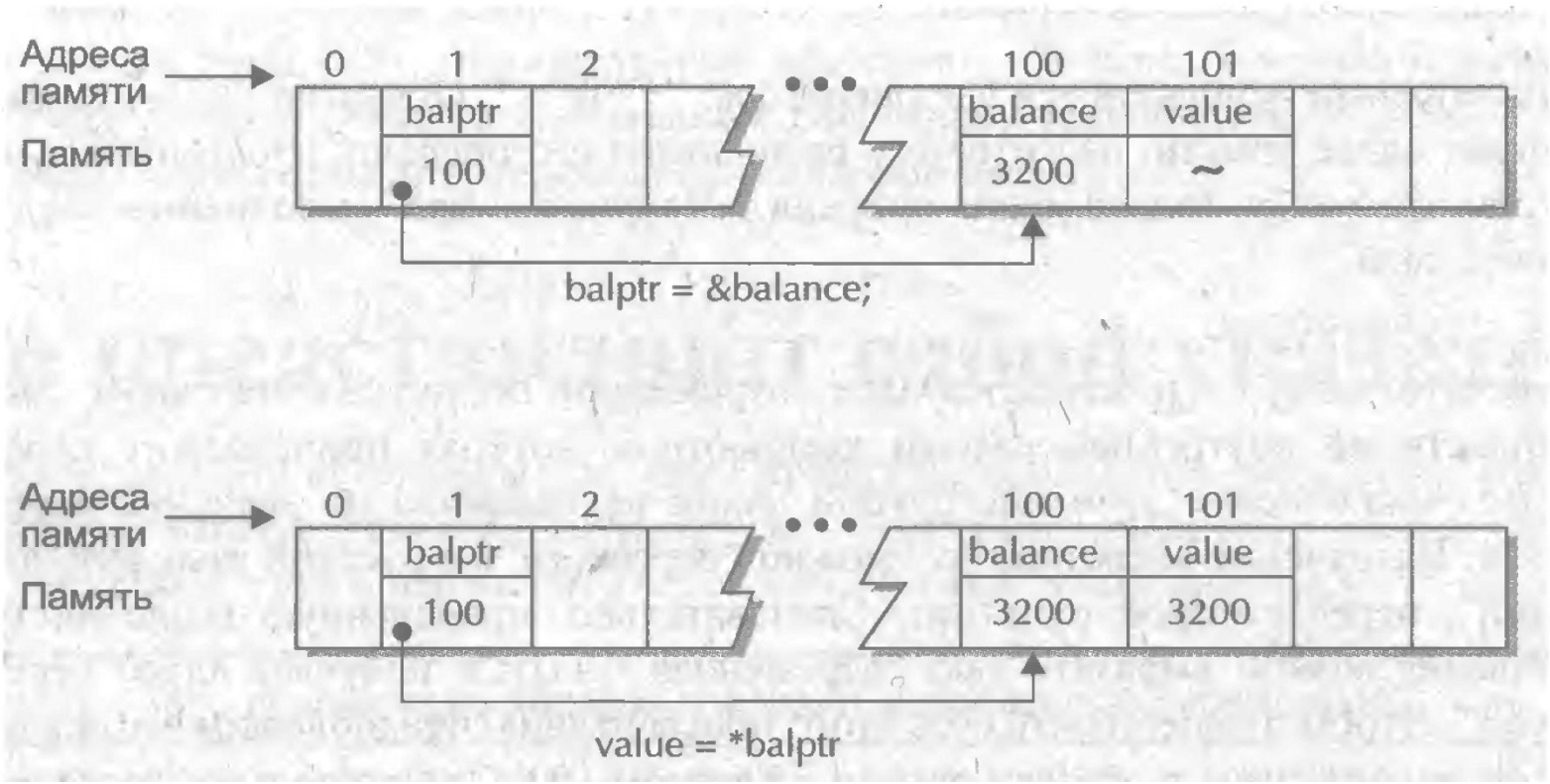
Оператор & возвращает адрес памяти, по которому расположен его операнд

```
balptr = &balance;
```

Оператор * обращается к значению переменной, расположенной по адресу, заданному его операндом.

```
value = *balptr;
```

```
#include <iostream>
using namespace std;

int main()
{
  int balance;
    int *balptr;
    int value;

    balance = 3200;
    balptr = &balance;
    value = *balptr;
    cout << "Баланс равен: " << value << '\n';

    return 0;
 }
```



Операторы * и & - операторы непрямого доступа

```
include <iostream>
sing namespace std;

nt main()

 char str[80];

 cout << "Введите строку: ";
 cin >> str; // Считываем строку с клавиатуры.
 cout << "Вот ваша строка: ";
 cout << str;

 return 0;
```

```
Введите строку: Это проверка
Вот ваша строка: Это
```

множества

```
#include <set>
set <int> s;
 s.insert(x);  добавление
s.erase(x);   удаление
s.find(x)       if (s.find(x) == s.end())
s.size();
```

Вывести всё содержимое множества можно двумя способами.

```
for (auto now = s.begin(); now != s.end(); now++) {   cout << *now << ' '; }
```

```
for (auto now : s) { cout << now << ' '; }
```

multiset

```
multiset <int> s;


int cnt = 0;
for (auto now = s.lower_bound(1);
now != s.upper_bound(1); now++)
{ cnt++; }
```

```
#include <iostream>
#include <set>
using namespace std;

int main()
{set <int> s;
multiset <int> s1;
int n, i, a, k;
cin>>n;
for(i=0; i<n; i++)
{
   cin>>a;
   s.insert(a);
   s1.insert(a);
}
k=s.size();
cout<<k<<endl;
for (auto now : s) { cout << now << ' '; }
cout<<endl;
for (auto now : s1) { cout << now << ' '; }
   return 0;
}
```

```
#include <iostream>
#include <set>
using namespace std;
int main()
{set <int> s;
int n, i, a, k;
cin>>n;
for(i=0; i<n; i++)
{
   cin>>a;
   if (s.find(a) == s.end())
      {cout<<"NO"<<endl;
       s.insert(a);
      }
   else
      cout<<"YES"<<endl;
}
   return 0;
}
```

Словари

```
#include <map>
map <int, string> s;
map <string, vector <string>> s;
s["Vasya"] = { "112133", "12341" };
```

```
map <int, string> s;
 s[112] = "sos";
s[102] = "emergency";
for (auto now : s)
{ cout << now.first << " " << now.second << "\n"; }
```

Функции

Прототипы функций

Три вида информации:

- тип возвращаемого ею значения;
- тип ее параметров;
- количество параметров.

Прототипы позволяют выполнить три операции:

- Они сообщают компилятору, код какого типа необходимо генерировать при вызове функции. Различия в типах параметров и значении, возвращаемом функцией, обеспечивают различную обработку компилятором.
- Они позволяют C++ обнаружить недопустимые преобразования типов аргументов, используемых при вызове функции, в тип, указанный в объявлении ее параметров, и сообщить о них.
- Они позволяют компилятору выявить различия между количеством аргументов, используемых при вызове функции, и количеством параметров, заданных в определении функции.

Общая форма прототипа:

```
type func_name(type parm_name1, type parm_name2,...,
               type parm_nameN);
```

Локальные переменные

Локальная переменная известна только той функции, в которой она определена.

Глобальные переменные

Глобальная переменная известна на протяжении всей программы, ее можно использовать в любом месте кода и она сохраняет свое значение во время выполнения всей программы.

```
#include <iostream>
using namespace std;

void func();

int main()
{
  int x; // Локальная переменная для функции main().

  x = 10;


  func();
  cout << "\n";
  cout << x; // Выводится число 10.

  return 0;
}

void func()
{
  int x; // Локальная переменная для функции func().

  x = -199;
  cout << x; // Выводится число -199.
}
```

```
#include <iostream>

using namespace std;
void func();
int a,b;
int main()
{
    cin >> a >> b;
    func();

    return 0;
}
```

```
void func()
{
    int c;
    c = a+b;
    cout << c;
}
```

```
#include <iostream>
using namespace std;
void func();
int a,b;
int main()
{
    cin >> a >> b;
    func();
cout << a << endl;
    return 0;
}
```

```
void func()
{ int a;
a = 10;
    int c;
    c = a+b;
    cout << c << endl;
}
```

```
2 5
15
2
```

Формальные параметры

Если функция использует параметры, то они должны быть объявлены. Эти переменные называются формальными параметрами.

```
#include <iostream>
using namespace std;
void func(int *i);
int a,b;
int main()
{int i;
    cin >> a >> b;
    func(&i);
cout << i << endl;
    return 0;
}
```

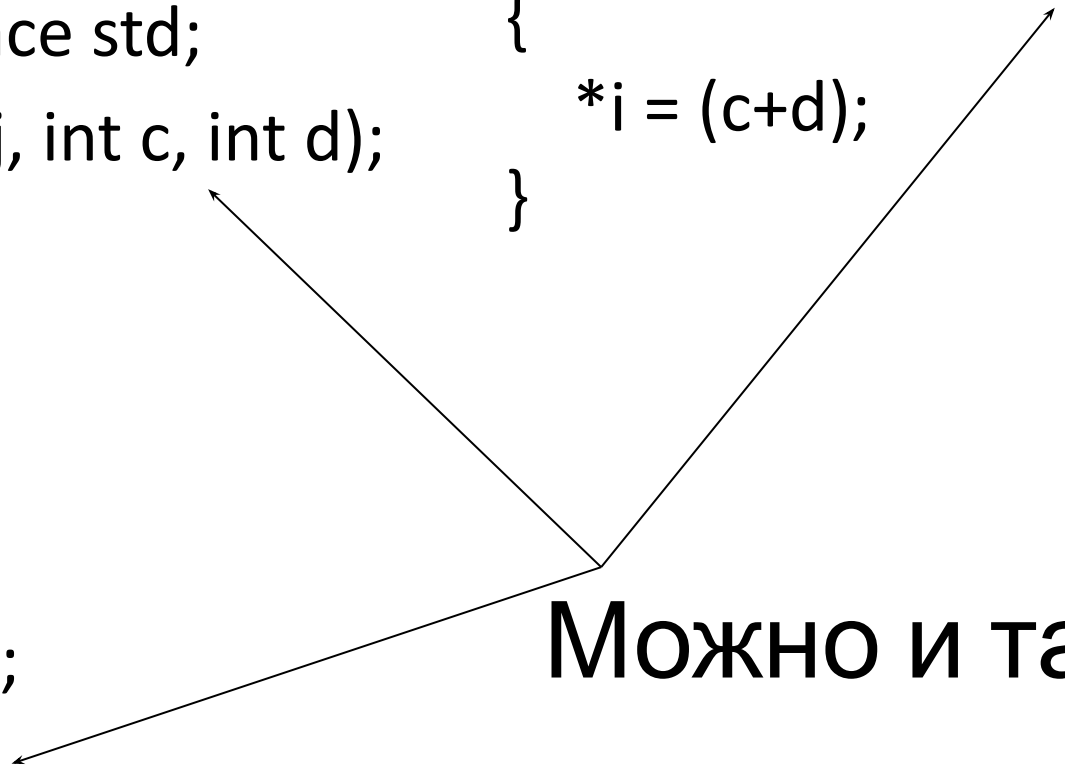
```
void func(int *i)
{
    *i = (a+b);
}
```

```
#include <iostream>
using namespace std;
void func(int *j, int *c, int *d);

int main()
{int i;
int a,b;
    cin >> a >> b;
    func(&i,&a,&b);
cout << i << endl;
    return 0;
}
```

```
void func(int *i, int *c, int *d)
{
    *i = (*c+*d);
}
```

```
#include <iostream>
using namespace std;
void func(int *j, int c, int d);

int main()
{int i;
int a,b;
    cin >> a >> b;
    func(&i,a,b);
cout << i << endl;
    return 0;
}
```

```
void func(int *i, int c, int d)
{
    *i = (c+d);
}
```



Можно и так

Ссылочные параметры

```
#include <iostream>
using namespace std;
void func(int &j);
int main()
{int i,ii;
    cin >> i;
    ii=i;
    func(ii);
cout << i << ' ' << ii << endl;
    return 0;
}
```

```
void func(int &i)
{
    i = 2*i;
}
```

Пример ввода-вывода:

5

5 10

Ограничения при использовании ссылок

- Нельзя ссылаться на ссылочную переменную.
- Нельзя создавать массивы ссылок.
- Нельзя создавать указатель на ссылку, т.е. нельзя к ссылке применять оператор “&”.
- Ссылки не разрешено использовать для битовых полей структур. (Битовые поля рассматриваются ниже в этой книге.)

Вызов функций с массивами

Функции передается только адрес первого элемента массива

1 способ

```
#include <iostream>
using namespace std;
void func(int mas[5]);
int main()
{int i;
int t[5];
for(i=0; i<5; i++) t[i]=i;
   func(t);
   return 0;
}
```

```
void func(int mas[5])
{
int i;
for(i=0; i<5; i++)
   cout << mas[i] << endl;

}
```

2 способ

```
#include <iostream>
using namespace std;
void func(int mas[]);
int main()
{int i;
int t[5];
for(i=0; i<5; i++) t[i]=i;
   func(t);
   return 0;
}
```

```
void func(int mas[])
{
int i;
for(i=0; i<5; i++)
   cout << mas[i] << endl;

}
```

3 способ

```
#include <iostream>
using namespace std;
void func(int *mas);
int main()
{int i;
int t[5];
for(i=0; i<5; i++) t[i]=i;
   func(t);
   return 0;
}
```

```
void func(int *mas)
{
int i;
for(i=0; i<5; i++)
   cout << mas[i] << endl;

}
```

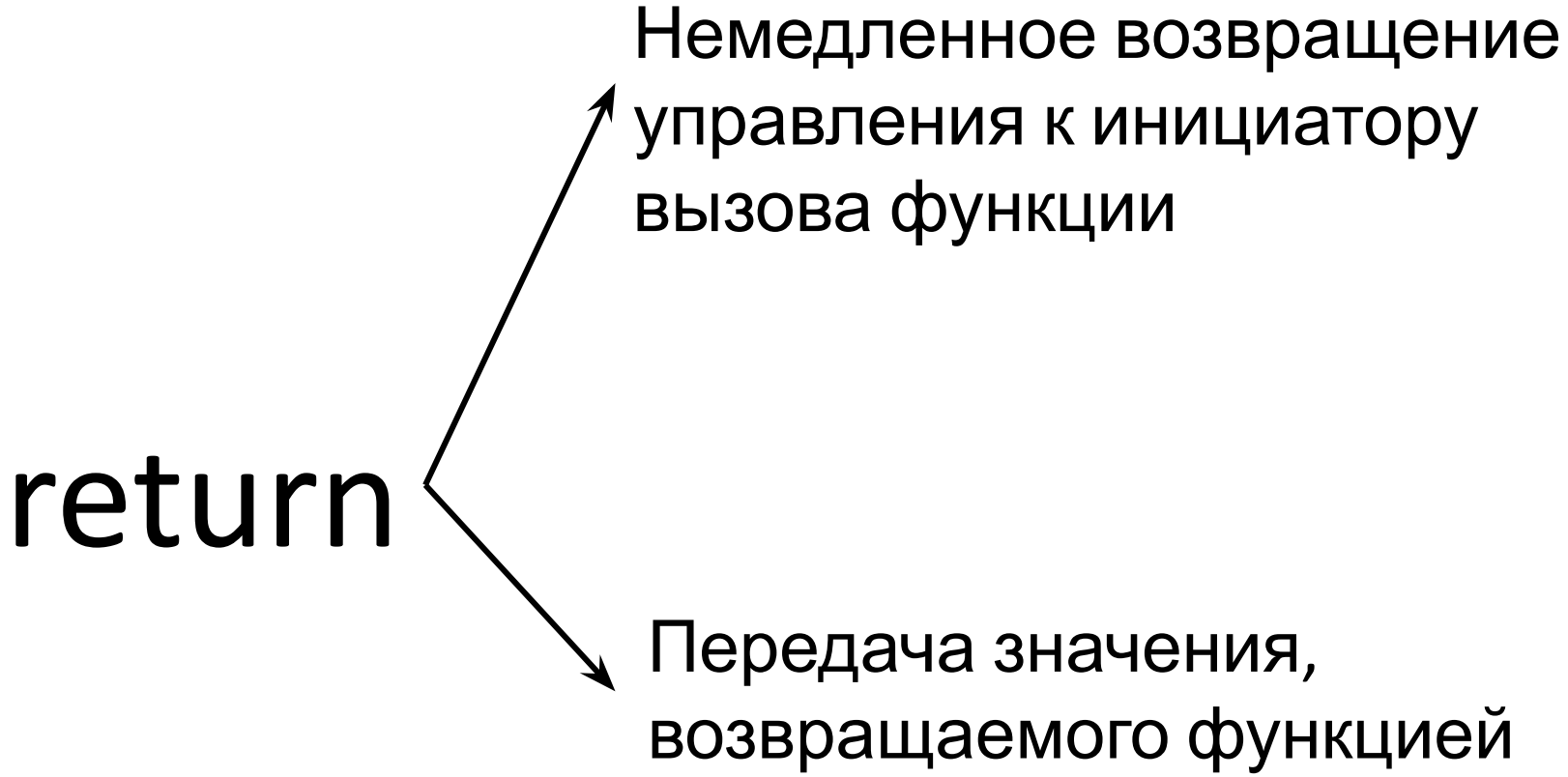
```
#include <iostream>
using namespace std;
void cube(int *mas, int num);
int main()
{int i; int m[10];
for(i=0; i<10; i++) m[i]=i;
for(i=0; i<10; i++)
cout <<m[i] << ' ';
cout << '\n';
    cube(m,10);
for(i=0; i<10; i++)
 cout <<m[i] << ' ';
    return 0; }
```

```
void cube(int *mas, int num)
{
int i;
for(i=0; i<10; i++)
   {
      *mas =*mas * *mas * *mas;
      mas++;
 //  mas[i]=mas[i]*mas[i]*mas[i];
   }
}
```

```
// C ++ программа для демонстрации работы
векторов
// можно передать по ссылке.
#include<bits/stdc++.h>
using namespace std;

// vect передается по ссылке и изменяется
// сделано здесь отражено в main ()
void func(vector<int> &vect)
{
   vect.push_back(30);
}

int main()
{
    vector<int> vect;
    vect.push_back(10);
    vect.push_back(20);

    func(vect);

    for (int i=0; i<vect.size(); i++)
       cout << vect[i] << " ";

    return 0;
}
```

```
#include <iostream>
#include <cstring>
#include <cctype>
using namespace std;
void stringupper(char *s);
int main()
{char str[80];
strcpy(str,"abcdefgh");
cout <<str << ' ';
stringupper(str);
cout <<str << ' ';
   return 0;
}
```

```
void stringupper(char *s)
{
while(*s)
  {
    *s =toupper(*s);
    s++;
  }
}
```


return
Немедленное возвращение управления к инициатору вызова функции
Передача значения, возвращаемого функцией

```
void power(int base, int exp)
{
   int i;
   if(exp<0) return; /* Чтобы не допустить возведения
                        числа в отрицательную степень,
                        здесь выполняется возврат в
                        вызывающую функцию и игнорируется
                        остальная часть функции. */
   i = 1;
   for( ; exp; exp--) i = base * i;
   cout << "Результат равен: " << i;
}
```

```
#include <iostream>
using namespace std;
int find_substr(char *sub, char *str);
int main()
{
   int index;
   index = find_substr("aaa","abc efgh aaa bbb");
   cout << index;
   return 0;
}
```

```
int find_substr(char *sub, char *str)
 {
  int  t;
  char *p, *p2;
  for(t=0; str[t]; t++)
  {
     p = &str[t];
     p2 = sub;
while(*p2  &&  *p2==*p)
     {
        p++;
        p2++;
     if(!*p2)  return t;
  }
  return -1;
 }
```

Структуры данных

```
#include <iostream>
#include <cstring>
using namespace std;
  struct one


    {
      string name;
      int w;
    };

int main()
{struct one ww[10];
int i;
string ss;
ss="aaa";
for(i=0; i<=5; i++)
{
    ww[i].w=i;
    ww[i].name=ss;
}
    for(i=0; i<=5; i++)
{
     cout << ww[i].w <<' ' <<
ww[i].name <<endl;

}

    return 0;
}
```

```
#include <cstdio>
#include <cstdlib>
using namespace std;
int main()
{int n=20;
int *a;
a=(int*)malloc(n*sizeof(int));
    system("pause");
    return 0;
}
```

```
int *a=new int[n];
```

```
int *doubleSize(int *arr, int
sz)
 { int *arr2 = new int[sz * 2];
 for (int i = 0; i < sz; i++)
arr2[i] = arr[i];
delete[] arr;
return arr2; }
```

//Генератор псевдослучайных чисел
#include <iostream> #include <stdlib.h> #include <time.h> using namespace std; int main() { int m; srand(time(NULL)); for(int i = 0; i < 10; i++) { m = 30 + rand() % 21; cout << m << endl; } return 0; }

Файлы

ifstream для чтения,

ofstream для записи

fstream для модификации файлов

#include <fstream>

```
ifstream file1;
file1.open("d2.txt");
 ofstream file2;
file2.open("dd2.txt");

file1.close();
file2.close();
```

Запись

```
file1 << buff << endl << vx << endl << p << endl;
```

Чтение

```
file1 >> buff >> vx >> p;
```

```
getline(file, s)
```

Открытие файлов

Режим	Назначение
in	Открыть для ввода (выбирается по умолчанию для ifstream)
out	Открыть для вывода (выбирается по умолчанию для ofstream)
binary	Открыть файл в бинарном виде
app	Присоединять данные; запись в конец файла
ate	Установить файловый указатель на конец файла
trunc	Уничтожить содержимое, если файл существует (выбирается по умолчанию, если флаг out указан, а флаги ate и app — нет)

```
ifstream file;
file1.open ("test.txt", ios::in | ios::binary);

ofstream file;
file1.open ("test.txt", ios::out | ios::app);
```

Запись

```
file1 << buff << endl << vx << endl << pi << endl;
```

Чтение

```
file1 >> buff >> vx >> pi;
```

Класс ifstream: чтение файлов

Метод	Описание
open	Открывает файл для чтения
get	Читает один или более символов из файла
getline	Читает символьную строку из текстового файла или данные из бинарного файла до определенного ограничителя
read	Считывает заданное число байт из файла в память
eof	Возвращает ненулевое значение (true), когда указатель потока достигает конца файла
peek	Выдает очередной символ потока, но не выбирает его
seekg	Перемещает указатель позиционирования файла в заданное положение
tellg	Возвращает текущее значение указателя позиционирования файла
close	Закрывает файл

```
ifstream file1("Temp.txt");
 char buff[100];
Int vx; float pi;
 file.getline(buff, sizeof(buff));
file >> vx >> pi:
```

Класс ofstream: запись файлов

Метод	Описание
open	Открывает файл для записи
put	Записывает одиночный символ в файл
write	Записывает заданное число байт из памяти в файл
seekp	Перемещает указатель позиционирования в указанное положение
tellp	Возвращает текущее значение указателя позиционирования файла
close	Закрывает файл

Бинарные файлы

```
#include <iostream>
#include <fstream>
 #include <locale>
using namespace std;
struct Notes { // структура данных записной книжки
char Name[60]; // Ф.И.О.
char Phone[16]; // телефон
int Age; // возраст };
int main()
{ setlocale(LC_ALL, "Russian");
Notes Note1= { "Грозный Иоанн Васильевич", "не установлен", 60 };
Notes Note2= { "Годунов Борис Федорович ", "095-111-2233 ", 30 };
Notes Note3= { "Романов Петр Михайлович ", "812-333-2211 ", 20 };
ofstream ofile("Notebook.dat", ios::binary);
ofile.write((char*)&Note1, sizeof(Notes)); // 1-й блок
 ofile.write((char*)&Note2, sizeof(Notes)); // 2-й блок
ofile.write((char*)&Note3, sizeof(Notes)); // 3-й блок
ofile.close();
// закрыть записанный файл
ifstream ifile("Notebook.dat", ios::binary);
Notes Note; // структурированная переменная
```

Класс fstream: произвольный доступ к файлу

```
ifstream ifile("Notebook.dat", ios::binary);
int pos = 49 * sizeof(Notes);
ifile.seekg(pos); // поиск 50-й записи
 Notes Note;  //Notes – описанная выше структура "запись"
ifile.read((char*)&Note, sizeof(Notes));
```

```
#include <iostream>
#include <fstream>
#include <locale>
using namespace std;
struct Notes { char Name[60]; char Phone[16];
int Age; };
 int main()
{ setlocale(LC_ALL, "Russian");
Notes Note1, Note3;
```

```
#include <iostream>
#include <set>
#include <map>
using namespace std;
int main()
{
   map <int, set<int>> s;
   map <int, set<int>> s1;
   int n, q, a,  b, i, k, l, p;
   cin>>n;
   for(i=0; i<n; i++)
   {
      cin>>a>>b;
      s[a].insert(b);
   }
   for(auto w: s)
   {
     cout << w.first << " " ;
     for(auto w1: w.second)
     cout<< w1 <<' ';
     cout<<endl;
   }
cout<<endl;
  s1[1]=s[1];
  for(auto w: s1)
   {
     cout << w.first << " " ;
     for(auto w1: w.second)
     cout<< w1 <<' ';
     cout<<endl;
   }
cout<<endl;
  s.erase(1);
  for(auto w: s)
   {
     cout << w.first << " " ;
     for(auto w1: w.second)
     cout<< w1 <<' ';
     cout<<endl;
   }

cout<<endl;
l=s.size();
cout<<l<<endl;
i=1;
  while(l>0)
  {
   i++;
   for(auto v: s1[i-1])
     {
       k=v;
       for(auto v1: s[k])
          s1[i].insert(v1);
       s.erase(k);
     }
    l=s.size();
   }

 for(auto w: s1)
   {
     cout << w.first << " " ;
     for(auto w1: w.second)
     cout<< w1 <<' ';
     cout<<endl;
   }
return 0;
}
```

НОД

```
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
    cout << "gcd(6, 20) = " << __gcd(6, 20)
<< endl;
}
```