

# Лекция 1

Алгоритмические языки и  
программирование

# Алгоритм

- Алгоритм — набор инструкций, описывающих *порядок* действий исполнителя для достижения некоторого результата.

# Программирование

- Программирование — процесс создания компьютерных программ.

# Компилятор

- **Компилятор** — программа, выполняющая преобразование файла с исходным кодом программы в исполняемый файл.
- **Компиляция** — преобразование программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке (машинном коде).
- Входной информацией для компилятора (исходный код) является описание алгоритма или программы на понятном человеку языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (машинный код).

# Введение в программирование и язык С

# Часть 1

# Препроцессор

- Препроцессор — преобразует исходный текст программы согласно указанным командам.
- `#include` — вставляет текст из указанного файла
- `#define` — задаёт макроопределение (макрос) или символическую константу
- `#undef` — отменяет предыдущее определение

# Типы данных

Целые типы	Размер в байтах (битах) x86/x64	Интервал изменения (для x86)
char	1 (8)/1(8)	от -128 до 127
unsigned char	1 (8)/1(8)	от 0 до 255
signed char	1 (8) /1(8)	от -128 до 127
int	2 (16) /4(32)	от -32768 до 32767
unsigned int	2 (16)/4(32)	от 0 до 65535
signed int	2 (16)/4(32)	от -32768 до 32767
short int	2 (16)/2(16)	от -32768 до 32767
unsigned short int	2 (16) /2(16)	от 0 до 65535
signed short int	2 (16)/2(16)	от -32768 до 32767
long int	4 (32)/8(64)	от -2147483648 до 2147483647
unsigned long int	4 (32) /8(64)	от 0 до 4294967295
signed long int	4 (32) /8(64)	от -2147483648 до 2147483647
Типы с плавающей запятой	Размер в байтах (битах)	Интервал изменения
float	4 (32)	от 3.4E-38 до 3.4E+38
double	8 (64)	от 1.7E-308 до 1.7E+308
long double	10 (80)	от 3.4E-4932 до 3.4E+4932

# Типы данных

- Общее правило размера данных согласно стандарту языка C:

char = 1 байт

char <= short int <= int <= long int

float = 4 байта

double в 2 раза больше float.

# Операторы

- Знак операции – это символ или группа символов, которые сообщают компилятору о необходимости выполнения определенных арифметических, логических или других действий.

# Операторы

Математические  
операторы

**Знак операции**

**Назначение**

\*

Умножение

/

Деление

%

Определение остатка от деления

+

Сложение

-

Вычитание

++

Увеличение на единицу(инкремент)

--

Уменьшение на единицу(декремент )

Операторы

присваивания:

**Знак операции**

**Назначение**

=

Присваивание

+=, -=, \*=, /=, %=

Составные операции присваивания  
(например,  $a *= b$   
(т.е.  $a = a * b$ ) и т.д.)

# Приоритет операций

Операторы	Выполняются
()	слева направо
-- ++	справа налево
* / %	слева направо
+ -	слева направо
< <= > >=	слева направо
== !=	слева направо
&&	слева направо
	слева направо
= += -= *= /= %=	справа налево
,	слева направо

# Библиотека языка С

- Библиотека С - это множество predetermined функций, предназначенное для использования в С-программах. Применение библиотеки С делает программирование более легким.

# stdio.h

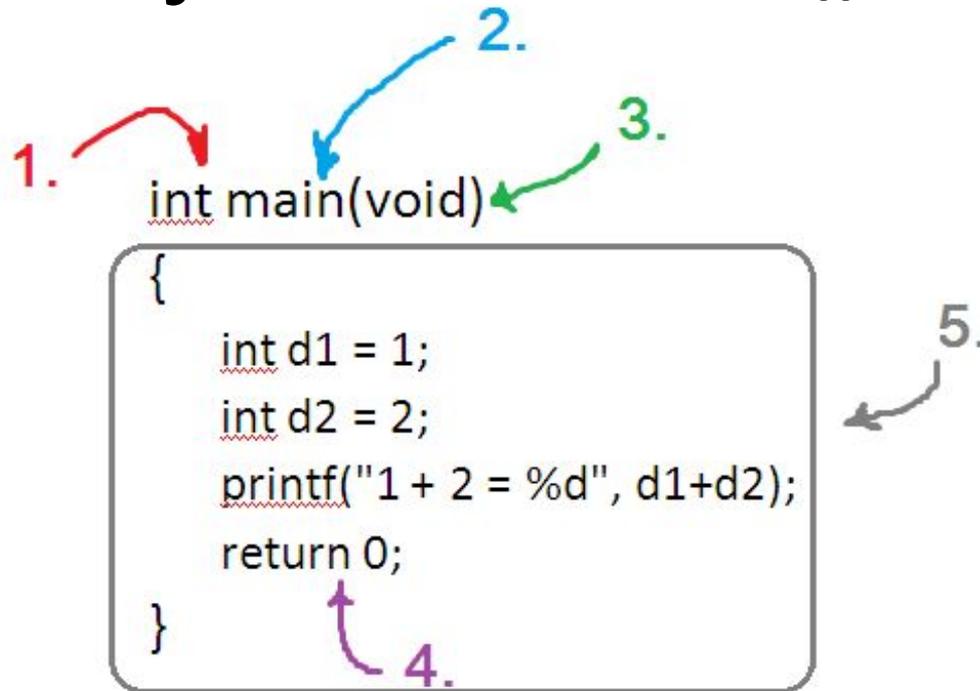
- Заголовочный файл стандартной библиотеки языка C.
- Содержит определения констант, объявления функций и типов для операций ввода/вывода.

# Функция main()

- Пример:

```
int main(void)
{
    int d1 = 1;
    int d2 = 2;
    printf("1 + 2 = %d", d1+d2);
    return 0;
}
```

# Функция main()



1. `int` – это тип возвращаемого функцией значения;

2. `main` — это имя функции;

3. `(void)` — это перечень аргументов функции. Слово `void` указывает, что у данной функции нет аргументов;

4. `return` – это оператор, который завершает выполнение функции и возвращает результат работы функции в точку вызова этой функции;

5. Часть функции после заголовка, заключенная в фигурные скобки называют телом функции.

# printf() и scanf()

- printf() – функция позволяет выводить на экран информацию согласно заданному формату.

```
int printf(char *format, arg1, arg2, ...);
```

- scanf() – функция позволяющая считывать с клавиатуры информацию согласно заданному формату.

```
int scanf(char *format, arg1, arg2, ...);
```

# Форматы printf и scanf

- %d – десятичный вывод (int)
- %c – вывод символа (char)
- %ld – десятичный длинный вывод (long int)
- %u – десятичный (unsigned int)
- %f или %g – дробный
- %6d – печатать в 6 позиций
- %.2f – печатать 2 знака после запятой
- %6.2f – печатать 2 знака после запятой в поле на 6 позиций

# Пример использования printf()

- char b;
  - int a;
  - printf(“Выведем число %d и букву %c”, a , b);
  - printf(“Выведем число %d и букву %c”, 3, ‘a’);
- 
- scanf(“%d”, &a); //ввод числа
  - scanf(“%c”, &b); //ввод буквы
  - scanf(“%d , %c”, &a, &b); //ввод числа и буквы

# Типы ошибок при разработке программ

- Синтаксические ошибки (syntax errors)
  - Забыли ; в конце выражения
  - Неправильно написали имя оператора
- Ошибки выполнения (runtime errors)
  - Деление на 0
  - Нет места при записи данных на диск
- Семантические (semantic errors)
  - Работает, но делает не то, что нужно.

# Здравствуй, мир!

```
#include <stdio.h>
int main (int argc, char *argv[])
{
    printf ("Hello, world!\n");
    return 0;
}
```

# Часть 2

# Кириллица в консоли

- В C есть функция `setlocale()`, которая выполняет перекодировку символов в соответствии с требуемым языком. Эта функция определена в заголовочном

файле `<locale.h>`

```
#include <stdio.h>
#include <locale.h>
int main()
{
    setlocale(LC_ALL, "Russian");
    printf("Русский текст в консоли");
    return 0;
}
```

# Циклы

- Иногда необходимо повторять одно и то же действие несколько раз подряд. Для этого используют циклы.
- Цикл - многократное прохождение по одному и тому же коду программы при выполнении заданного условия.

# ЦИКЛЫ

```
// Инструкция for  
for (выр1; выр2; выр3)  
    инструкция
```

```
// эквивалентна конструкции  
выр1;  
while (выр2)  
{  
    инструкция  
    выр3;  
}
```

# Примеры

## Цикл while и for

```
int i = 0;
while (i < 10) {
    printf("%d\n", i);
    i++;
}
```

1. *выр1*

2. *выр2*

3. *выр3*

```
int i;
for (i = 0; i < 10; i++) {
    printf("%d\n", i);
}
```

# Циклы

## Цикл While (с пост условием)

**do**

инструкция

**while** (выражение) ;

# Примеры

## Цикл While (с пост условием)

```
int i = 0;  
do {  
    printf("%d\n", i);  
    i++;  
} while(i < 10);
```

# Конструкция if-else

```
if (выражение)  
    инструкция1  
else  
    инструкция2
```

# Конструкция if-else

```
int a = 4, b = 3, z;
```

```
if (a > b)
```

```
    z = a;
```

```
else
```

```
    z = b;
```

```
// Переменной z присваивается  
большее значение из a или b.
```

# Вложенный if-else

```
int a = 4, b = 3, n = 1, z;  
if (n > 0)  
{  
    if (a > b)  
        z = a;  
}  
else  
    z = b;
```

# Конструкция else-if

**if** (выражение)

инструкция

**else if** (выражение)

инструкция

**else if** (выражение)

инструкция

**else if** (выражение)

инструкция

**else**

инструкция

# Пример

```
int a = 4, b = 3, z;  
  if (a > b)  
      z = a;  
  else if (a < b)  
      z = b;  
  else  
      z = 0;
```

**Что будет если a = 3?**

# Часть 3

# Управляющие символы

- Управляющие символы (или как их ещё называют — **escape-последовательность**) — символы которые выталкиваются в поток вывода, с целью форматирования вывода или печати некоторых управляющих знаков.

# Управляющие символы

## Экранирующие последовательности

- `\n` – new line
- `\r` – возврат каретки
- `\t` – табуляция
- `\b` – backspace
- `\"` и `\'` – кавычки
- `\\` – символ `\`
- `\a` – alarm (звуковой сигнал)

# Математические функции

- В `<math.h>` описано множество математических функций. Входные и выходные данные типа **double**.

**sin(x)**      синус  $x$ ,  $x$  в радианах.

**cos(x)**      косинус  $x$ ,  $x$  в радианах.

**atan2(y, x)** арктангенс  $y/x$ ,  $y$  и  $x$  в радианах.

**exp(x)**      экспоненциальная функция  $e^x$ .

**log(x)**      натуральный (по основанию  $e$ ) логарифм  $x$  ( $x > 0$ ).

**log10(x)**    обычный (по основанию 10) логарифм  $x$  ( $x > 0$ ).

**pow(x, y)**    степень  $x^y$ .

**sqrt(x)**      корень квадратный  $x$  ( $x > 0$ ).

# Лабораторные работы

# Математическое выражение

- Запрограммировать следующее выражение:

$$\left(a + b - \frac{f}{a}\right) + f * a - (a + b)$$

- Числа  $a$ ,  $b$ ,  $f$  вводятся с клавиатуры. Организовать пользовательский интерфейс, таким образом, чтобы было понятно, в каком порядке должны вводиться числа.
- Результат:

Введите цифры  $a$ ,  $b$  и  $f$ :

$a=15$

$b=10$

$f=3$

Вычисляем по формуле: `_(написать формулу тут)_`

$x=...$

# Математическое выражение

- Запрограммировать следующее выражение:

$$\left( a^{\frac{f}{2}} + b - \frac{f}{a} \right) + f * a^2 - (a^2 + b^2)$$

- Числа  $a$ ,  $b$ ,  $f$  вводятся с клавиатуры. Организовать пользовательский интерфейс, таким образом, чтобы было понятно, в каком порядке должны вводиться числа.
- Результат:

Введите цифры  $a$ ,  $b$  и  $f$ :

$a=15$

$b=10$

$f=3$

Вычисляем по формуле: `_(написать формулу тут)_`

$x=418$

# Деление числа на разряды

- Составить программу, которая будет считывать введённое пятизначное число. После чего, каждую цифру этого числа необходимо вывести в новой строке.
- Результат:

Деление числа на разряды

Введите пятизначное число: 10819

1 цифра равна 1

2 цифра равна 0

3 цифра равна 8

4 цифра равна 1

5 цифра равна 9

# Деление числа на

## • Подсказка **разряды**

1. Чтобы узнать последнюю цифру числа, нужно взять остаток от деления нашего пятизначного числа на 10.
2. Выводим число.
3. Затем делим наше пятизначное число на 10, чтобы сдвинуться на одну цифру.

$$\begin{array}{r} \begin{array}{r} -35191 \mid 10 \\ \underline{30} \\ -51 \\ \underline{50} \\ -19 \\ -10 \\ \underline{-91} \\ -90 \\ \underline{1} \end{array} \quad \begin{array}{r} -3519 \mid 10 \\ \underline{30} \\ -51 \\ \underline{50} \\ -19 \\ -10 \\ \underline{9} \end{array} \quad \begin{array}{r} -351 \mid 10 \\ \underline{30} \\ -51 \\ \underline{50} \\ 1 \end{array} \quad \begin{array}{r} -35 \mid 10 \\ \underline{30} \\ 5 \mid 3 \end{array} \end{array}$$

- Если число не пятизначное, вывести ошибку и приглашение повторить ввод.

# Конвертер мер длины

- Программа должна переводить число, введенное с клавиатуры, в выбранную меру длины. (1 миля = 1.609344 км)
- Пример:

Выберите режим работы программы:

1) Мили -> километры

2) Километры -> мили

>1

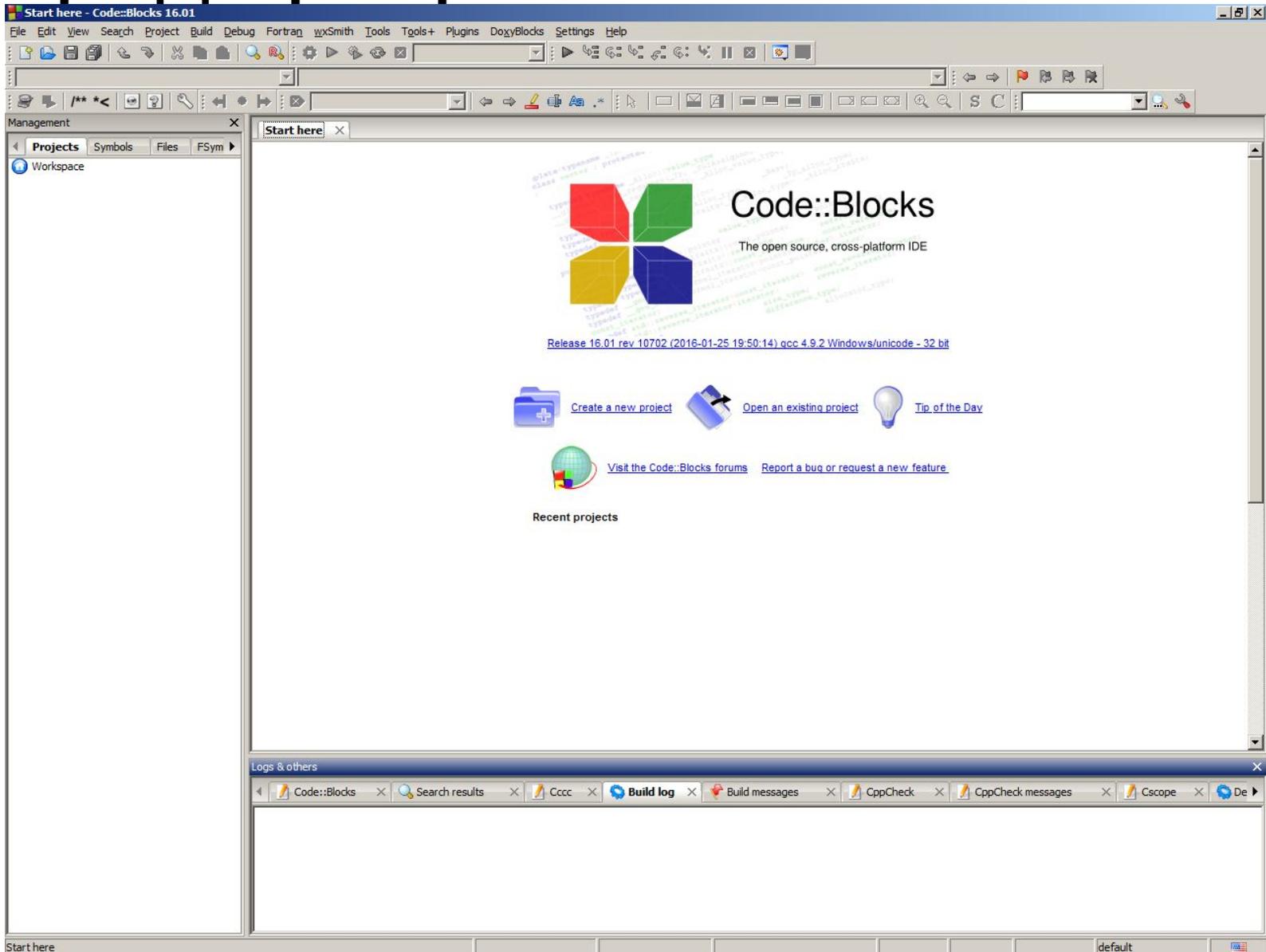
Введите количество Миль: 15

15 миль = 24.14016 километра

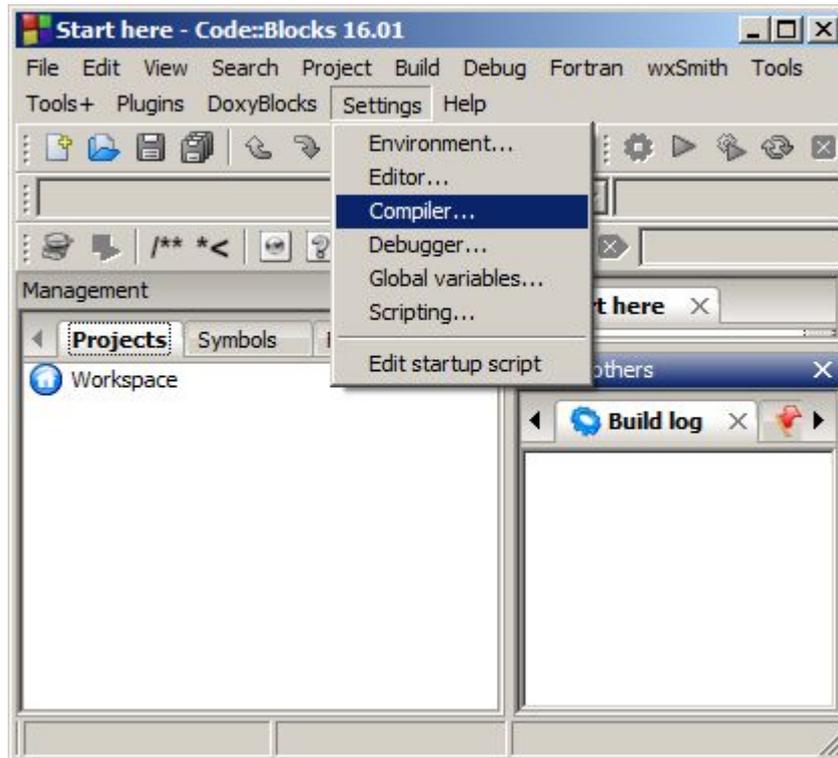
# Установка Code::Blocks

- <http://cppstudio.com/post/2650/>

# Среда разработки Code::Blocks

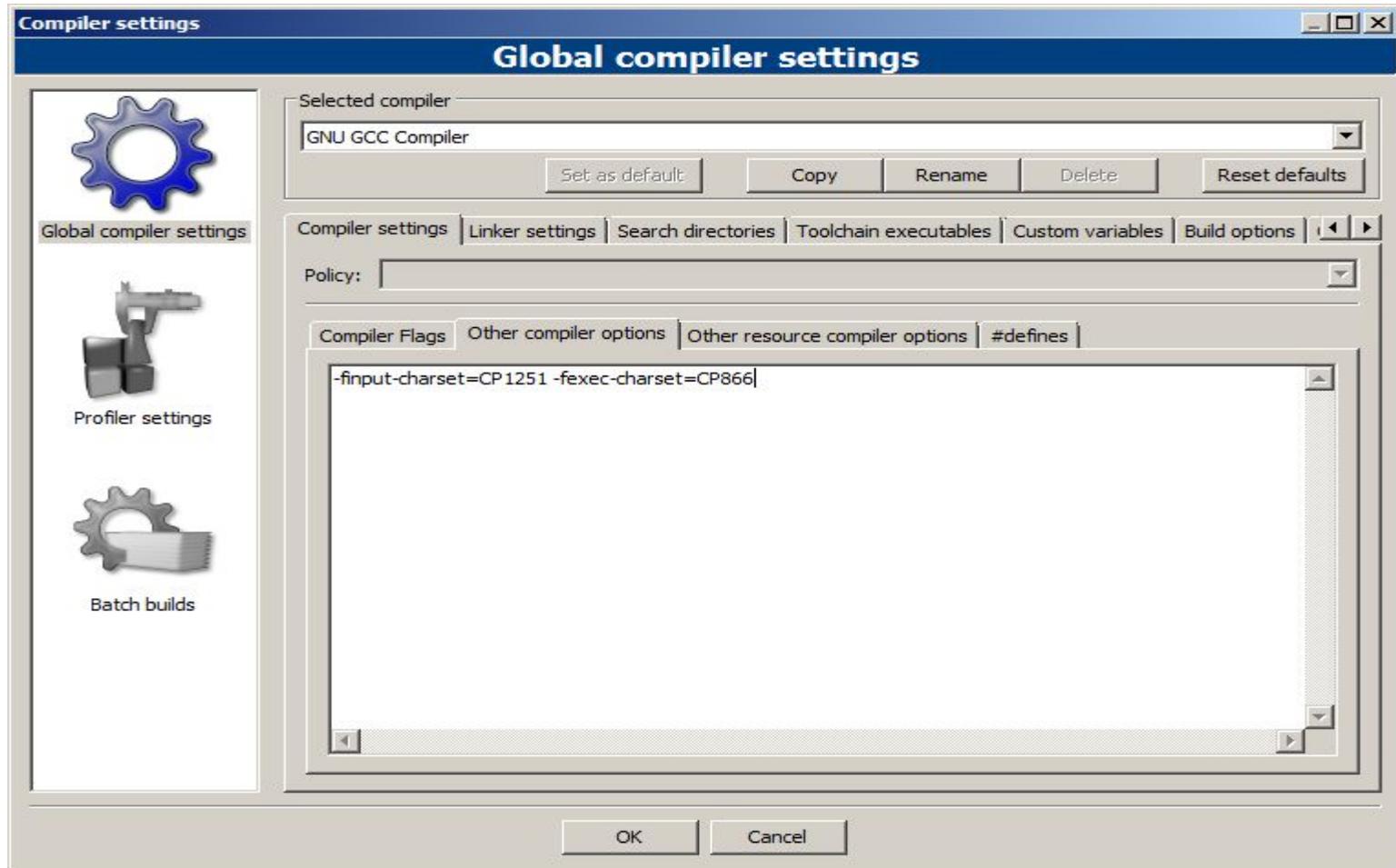


# Среда разработки Code::Blocks

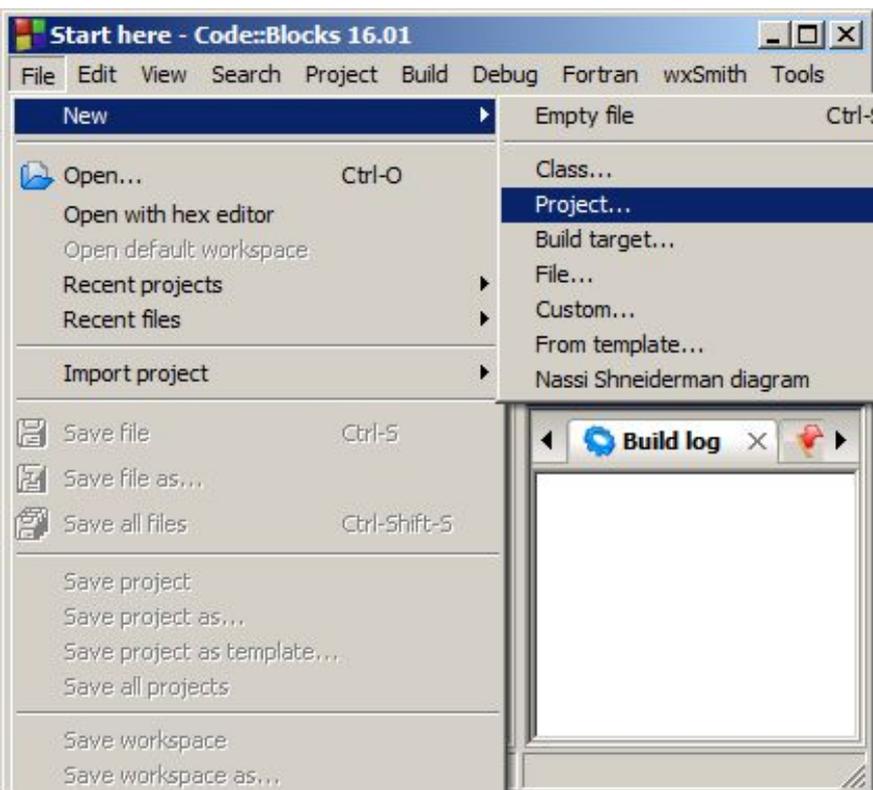


# Среда разработки Code::Blocks

-finput-charset=CP1251 -fexec-charset=CP866



# Среда разработки Code::Blocks



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

# Среда разработки Code::Blocks



# Среда разработки Code::Blocks

