Программирование
Программирование и алгоритмизация
Основы программирования
Введение в специальность

Лекция 3.

Си – простые типы данных, развилки, цикл DO WHILE, БЛОК СХЕМЫ!

Что будем изучать на курсе?

1) Основы основ – Си, VS, if, do while
2) Простейшая (статическая) графика в WinAPI. (Функции, рекурсия)
3) Динамическая графика (игры) – (циклы, switch, случайные числа, таймер)
4) Массивы – одномерные и двухмерные. В играх – естественно.
5) Структуры, файлы, строки (в том числе и в играх)
6) * Память, указатели, динамическая память.

Основные типы данных

Тип	Длина байт	Диапазон значений	Минимальное значение	Максимальное значение
char	1	2^8 = 256	-128	+127
short	2	**2^16= 65 536**	**-32 768**	32767
long	4	**2^32 = +4,294,967,296**	**−2,147,483,648**	**+2,147,483,647**
float	**4**	**IEEE 754 standard**	$1.175494351 \times 10^{-38}$	3.4028234×10^{38}
double	**8**	**IEEE 754 standard**	**2.22507e-308**	**1.79769e+308**

int = short/long

https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D0%BE%D0%B2_%D0%A1%D0%B8

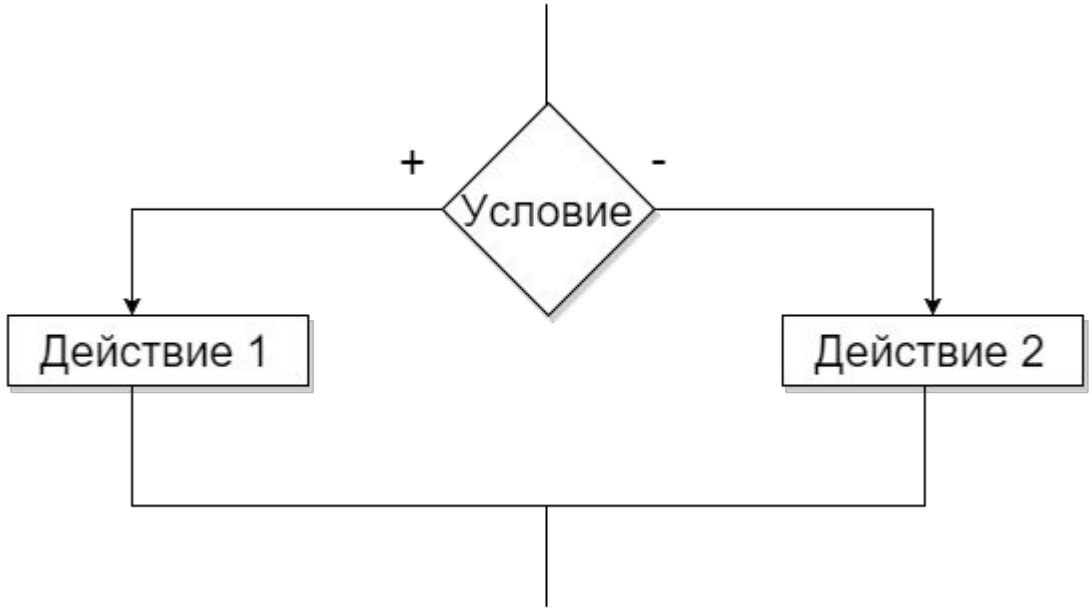
Строка форматирования

Тип	scanf/printf
char	%c
short	%hi
int	%d или %i
long	%li
float	%f
double	%lf
long double	%Lf

http://youngcoder.ru/lessons/3/formatnyi_vyvod_printf.php
http://www.c-cpp.ru/content/printf
https://ru.wikipedia.org/wiki/Printf

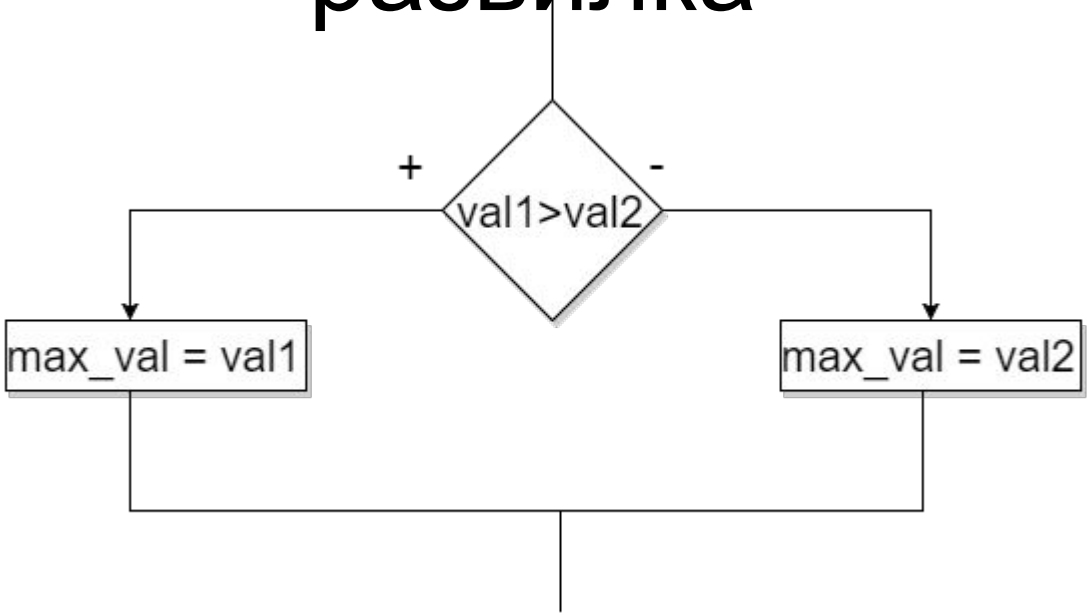
Консоль – что из себя представляет.
Знакоместо – что это такое.

Развилка (if)




```
if (Условие)
        Действие1;
else
        Действие2;
```

Найти максимум - полная развилка

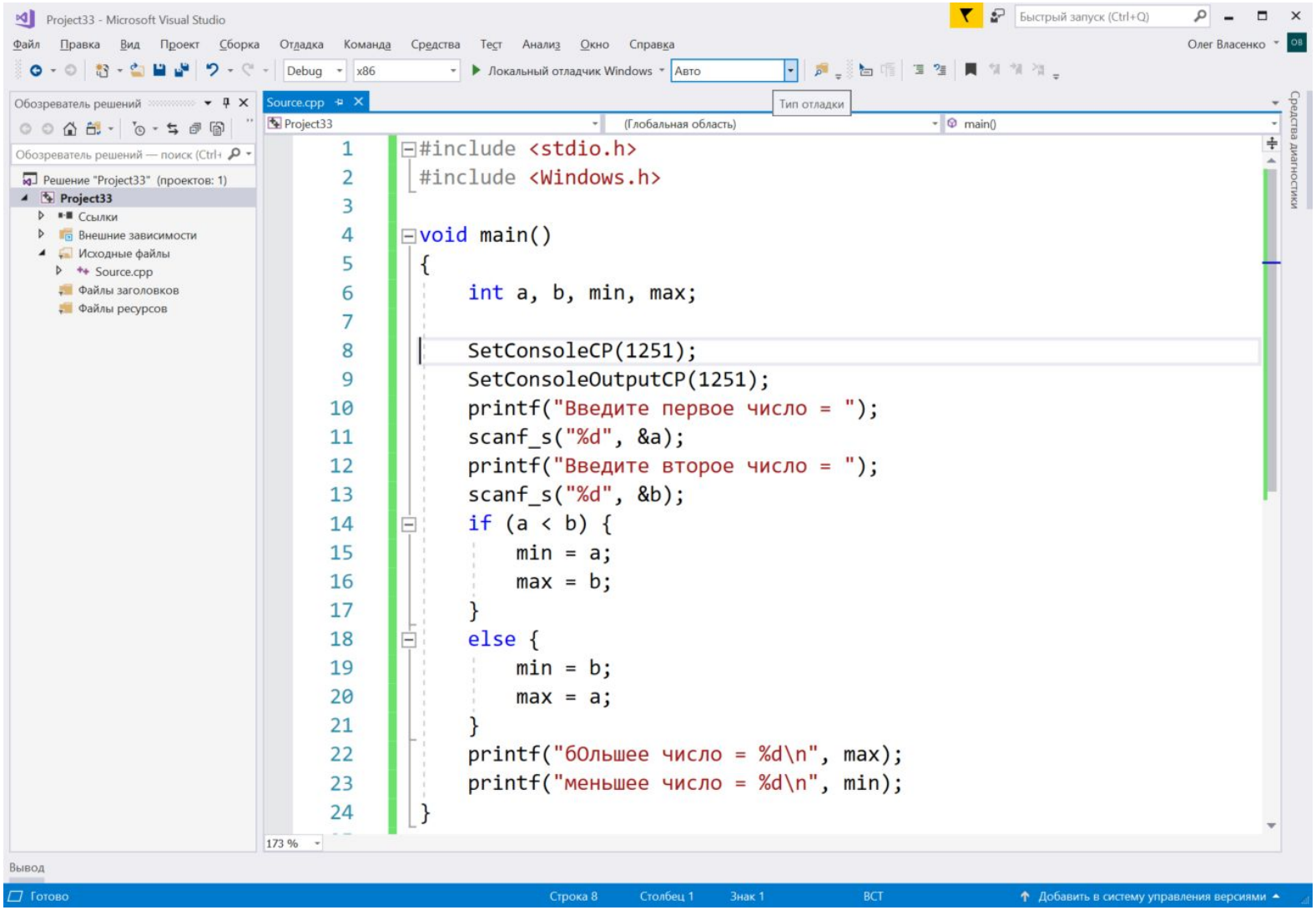


Фрагмент кода:

```
if (val1 > val2) {
      max_val = val1;
} else  {
      max_val = val2;
}
```

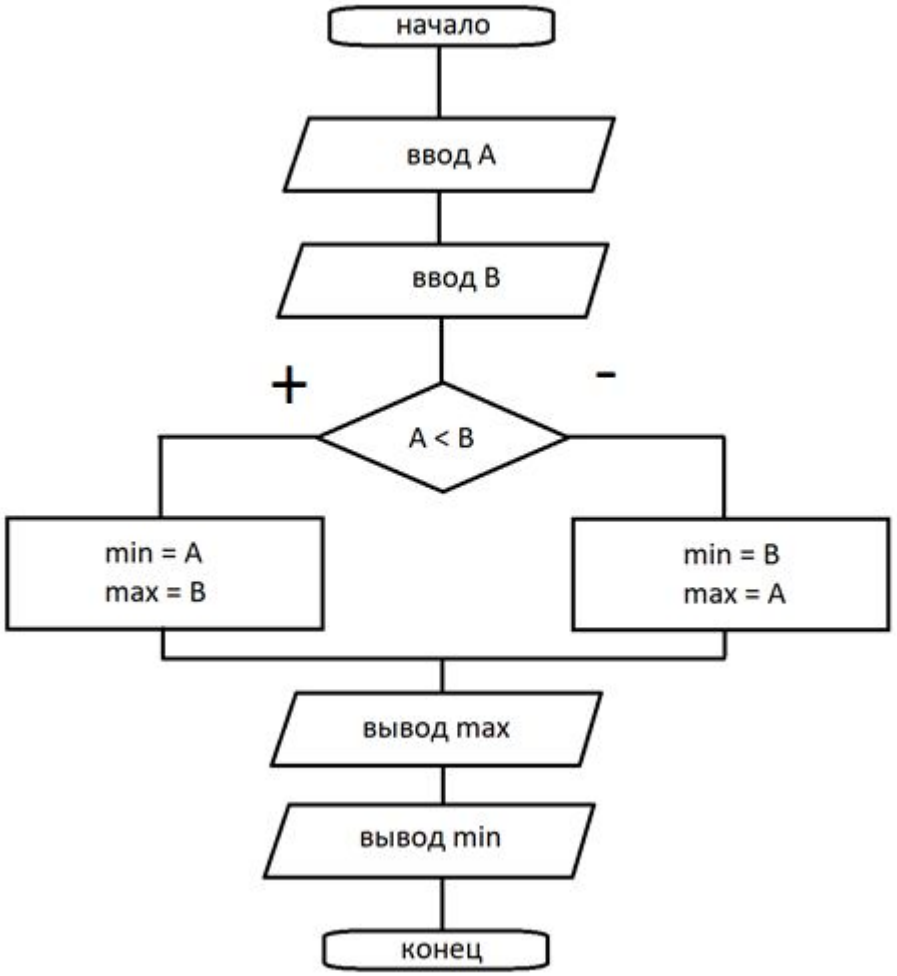
Задача. Какое число больше? (1)

Ввести два числа. Вывести бОльшее число. И вывести меньшее число.

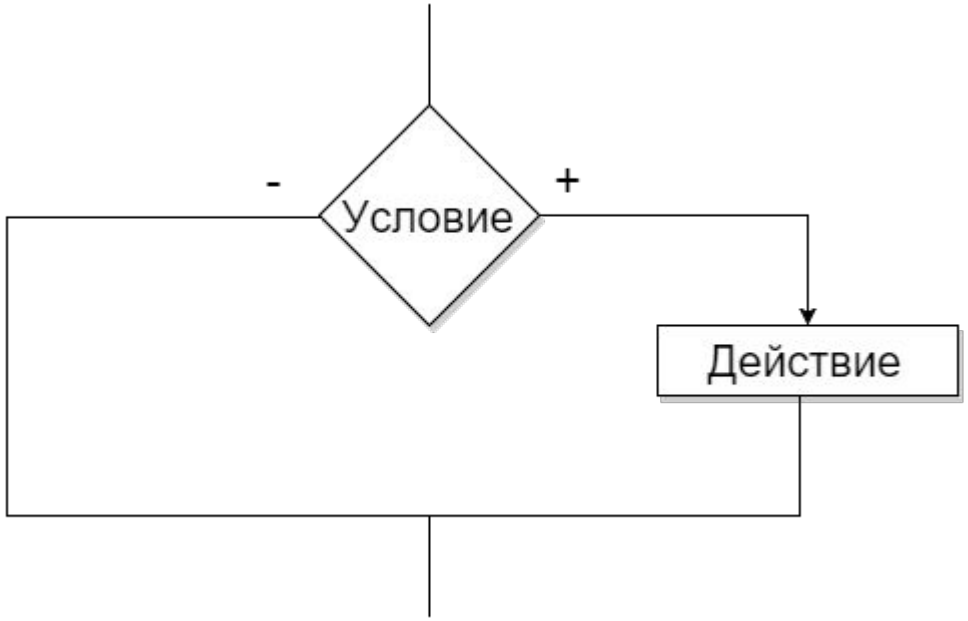


Задача. Какое число больше? (2)

Блок схема для программы

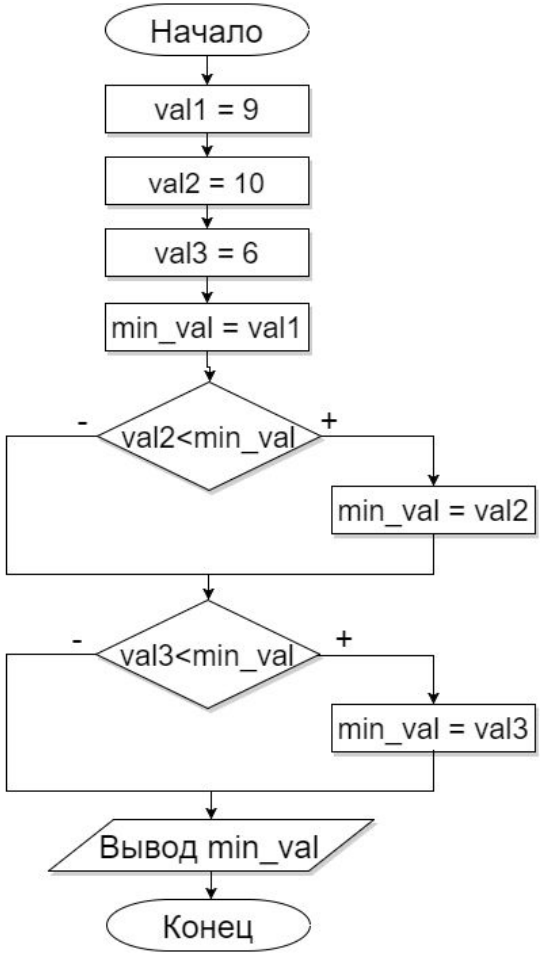


Усеченная развилка




```
if (Условие) {
    Действие;
}
```

Минимум из 3 чисел




```
void main() {

   int val1 = 9;
   int val2 = 10;
   int val3 = 6;

   int min_val = val1; // берем за минимальный  val1

   if (val2 < min_val) { // если второе меньше
      min_val = val2; // то теперь минимальное val2
   }

   if (val3 < min_val) { // если третье меньше
      min_val = val3; // то теперь минимальное val3
   }

   printf("min_val = %i", min_val);
}
```

Логические операции

Оператор	Описание
&&	Логическое И (AND)
||	Логическое ИЛИ (OR)
!	Логическое унарное НЕ (NOT)

A	!A
0	1
1	0

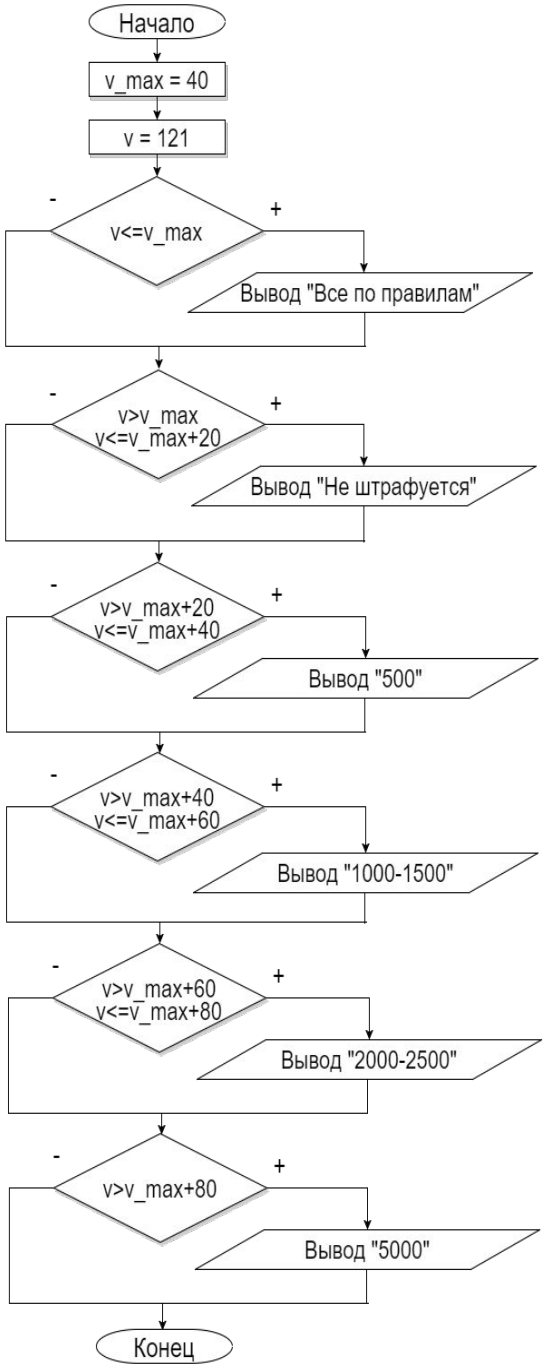
A	B	A && B	A || B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

```
if (time < 7.00 || day >= 6)  rest();

if (!closed && money > 1000)  eat();
```

Штраф за превышение скорости

```
void main() {
  int v_max = 40; // Максимально разрешенная скорость
  int v = 70; // Реальная скорость!

  if (v <= v_max) {
    printf("Все по правилам!");
  }
  if ((v > v_max) && (v <= v_max + 20)) {
   printf("не штрафуется");
  }
  if ((v > v_max + 20) && (v <= v_max + 40)) {
    printf("500");
  }
  if ((v > v_max + 40) && (v <= v_max + 60)) {
     printf("1000-1500");
  }
  if ((v > v_max + 60) && (v <= v_max + 80)) {
     printf("2000-2500");
  }
  if (v > v_max + 80) {
      printf("5000");
  }
}
```

Блок схема


Начало
v_max = 40
v = 121
v<=v_max
-
+
Вывод "Все по правилам"
v>v_max
v<=v_max+20
-
+
Вывод "Не штрафуется"
v>v_max+20
v<=v_max+40
-
+
Вывод "500"
v>v_max+40
v<=v_max+60
-
+
Вывод "1000-1500"
v>v_max+60
v<=v_max+80
-
+
Вывод "2000-2500"
v>v_max+80
-
+
Вывод "5000"
Конец

Штраф за превышение скорости - тесты

тесты, чтобы проверить все ситуации (при разрешенной скорости 40 км/час):

№	Что выводим	Вход
1	Все по правилам	38
2	Не штрафуется	59
3		
4		
5		
6		

Зачем нужны блок схемы?

1) Зачем учиться рисовать блок схемы, если их нельзя скомпилировать и выполнить? В них нельзя автоматически найти ошибки?
2) Зачем учиться рисовать блок схемы, если программа длиннее 100 строк ужасно выглядит в виде блок схемы и абсолютно нечитаема?
3) И зачем тратить наше бесценное время?