

# Программирование 1

## Лекция 2

Введение в язык программирования  
C++.

Стандартные типы данных.

Основные управляющие структуры  
(начало).

Внимание!

В начале лекции было завершено рассмотрение материала первой лекции

# Концепция данных. Типы данных

*Программы = структуры данных + алгоритмы*

*Компьютер - тоже состоит из структур данных (СД) и алгоритмов.*

*Компьютер - это интегрированный набор структур данных и алгоритмов, способный хранить и выполнять программы.*

*Встроенные СД = регистры и слова памяти, где хранятся двоичные коды (числа, символы и т.д.)*

*Встроенные в аппаратуру алгоритмы = воплощенные в электронных логических цепях (схемах) жесткие правила интерпретации данных в памяти как команд, подлежащих исполнению.  
(Это некоторое упрощение)*

# Данные в прикладных (предметных) областях

Задачи, которые должны решаться людьми с помощью компьютеров, формулируются и решаются не на языке битов,

а с помощью оперирования

*числами, литерами (буквами), текстами, символами,*

а также более сложными структурами, например,

*последовательностями, списками, деревьями, графами*

и т.п.

Далее идут *геометрические образы, изображения, видео*  
*и аудио образы* и т.п.

## Типы и структуры данных

Обеспечивают связь между двоичными (машинными) данными и прикладными («человеческими») данными.

**Языки высокого уровня** (в отличие от машинных языков) абстрагируются от деталей внутреннего машинного представления данных.

В программах используются **переменные**.

**Переменная = имя + значение.**

**Тип данных =**

- **множество значений**, которые могут принимать переменные и
- **набор операций**, которые можно к ним применять.

# Типы C++

Фундаментальные  
(стандартные)

Арифметические

Вещественные (с  
плавающей точкой)  
(float)

Целые  
(int)

Интегральные

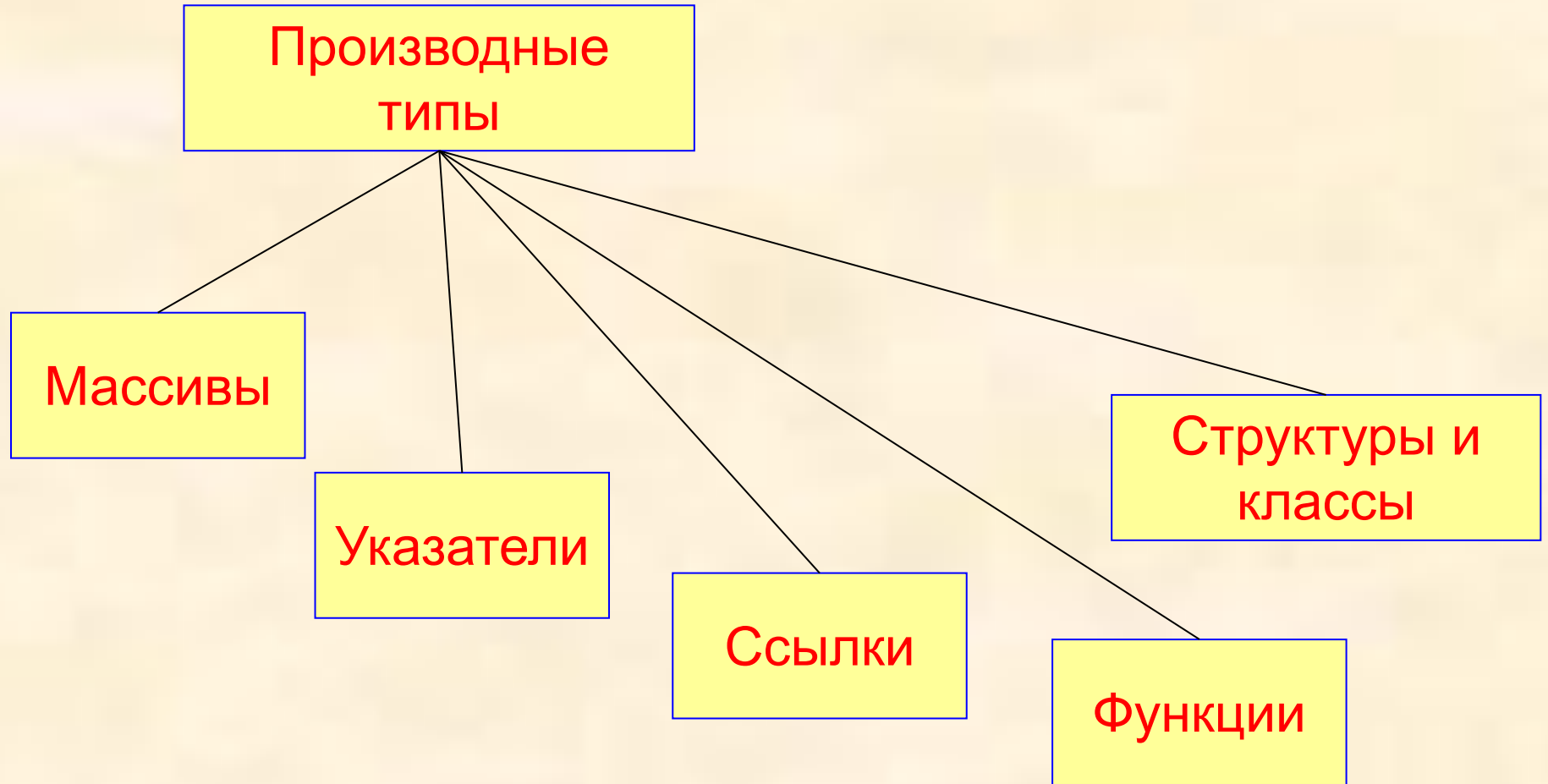
Символьные  
(char)

void

Перечисления\*  
(enum)

Логические  
(bool)

\* Определяется  
программистом, остальные  
типы - встроенные



Для описания стандартных типов определены следующие ключевые слова:

`int` (целый);

`char` (символьный);

`wchar_t` (расширенный символьный);

`bool` (логический);

`float` (вещественный);

`double` (вещественный с двойной точностью).

**Спецификаторы** типа, уточняющих внутреннее представление и диапазон значений стандартных типов:

`short` (короткий);

`long` (длинный);

`signed` (со знаком);

`unsigned` (без знака).



# Диапазоны значений простых типов данных

Тип	Диапазон значений	Размер (байт)
Bool	true и false	1
signed char	-128 ... 127	1
Unsigned char	0 ... 255	1
signed short int	-32 768 ... 32 767	2
Unsigned short int	0 ... 65 535	2
signed long int	-2 147 483 648 ... 2 147 483 647	4
Unsigned long int	0 ... 4 294 967 295	4
Float	$3.4e-38$ ... $3.4e+38$	4
Double	$1.7e-308$ ... $1.7e+308$	8
long double	$3.4e-4932$ ... $3.4e+4932$	10

# Стандартные арифметические операции +, -, \*, /, %

## Знаки отношений порядка

Равно (=)	==
Не равно ( $\neq$ )	!=
Меньше (<)	<
Больше (>)	>
Меньше или равно ( $\leq$ )	<=
Больше или равно ( $\geq$ )	>=

# Логический (булевский) тип

Джордж Буль (1815-1864)

Тип `bool`.

*Множество значений = {true, false}.*

Для краткости в свободном тексте (не в тексте программы) будем обозначать

*true = T, false = F.*

Константы, переменные, выражения булевского типа.

Определение булевского выражения см. далее.

Кроме записи булевских выражений в программах нам потребуется запись **утверждений**.

(Синтаксически **утверждения** представляются как комментарии)

**Утверждение = Высказывание.**

**Формула** - обозначение высказывания.

Высказывание (формула) образовано из других высказываний (**подформул**), в т.ч. «простейших».

# Логические операторы (логические связки):

(Словесное обозначение, в Паскале, в C++, в мат. логике)

Отрицание - не, not, !, ¬

Дизъюнкция - или, or, ||, ∨

Конъюнкция - и, and, &&, & или ∧

## Таблица истинности

$a$  и  $b$  - переменные (или другие «объекты» программы, принимающие булевские значения).

$Val(a)$  или короче  $V(a)$  - значение  $a$ .

$V(a)$	$V(b)$	$V(a \vee b)$	$V(a \& b)$	$V(\neg a)$
$F$	$F$	$F$	$F$	$T$
$F$	$T$	$T$	$F$	
$T$	$F$	$T$	$F$	$F$
$T$	$T$	$T$	$T$	

# Свойства булевских операций (проверяются по таблице истинности)

Коммутативность (перестановочный закон):

$$a \vee b = b \vee a,$$

$$a \& b = b \& a.$$

Ассоциативность (сочетательный закон):

$$(a \vee b) \vee c = a \vee (b \vee c),$$

$$(a \& b) \& c = a \& (b \& c).$$

Дистрибутивность (распределительный закон):

$$(a \vee b) \& c = (a \& c) \vee (b \& c),$$

$$(a \& b) \vee c = (a \vee c) \& (b \vee c).$$

## Законы Де Моргана:

$$\neg (a \vee b) = (\neg a) \& (\neg b),$$

$$\neg (a \& b) = (\neg a) \vee (\neg b).$$

## Полезные тождества

$$a \vee a = a,$$

$$a \& a = a.$$

$$a \vee T = T,$$

$$a \& T = a.$$

$$a \vee F = a,$$

$$a \& F = F.$$



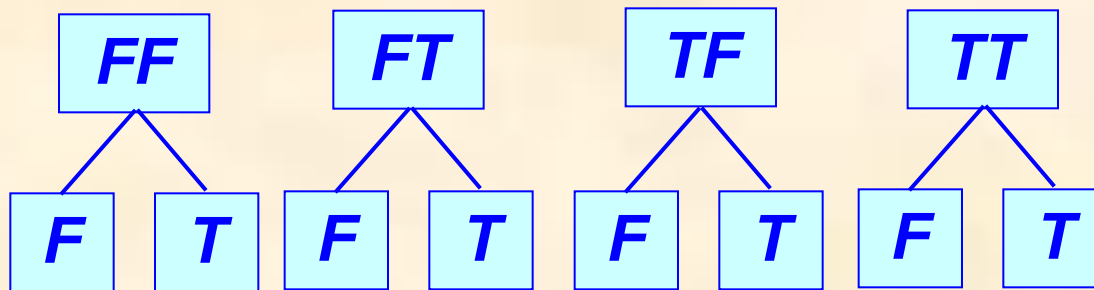
# Исключающее или (xor)

$V(a)$	$V(b)$	$V(a \text{ xor } b)$
$F$	$F$	$F$
$F$	$T$	$T$
$T$	$F$	$T$
$T$	$T$	$F$

$$a \text{ xor } b = (a \ \& \ \neg b) \vee (\neg a \ \& \ b)$$

# Всего можно определить 16 бинарных функций $a$ от $b$

Вариантов значений аргументов -  $2*2 = 4$   
 $(a = F, a = T) \otimes (b = F, b = T) \rightarrow FF, FT, TF, TT$



Каждый вариант может иметь 2 результата ( $F$  или  $T$ )  
Всего  $2*2*2*2 = 2^4 = 16$  различных четверок ответов.

Оказывается, что все 16 бинарных функций  
выражаются через

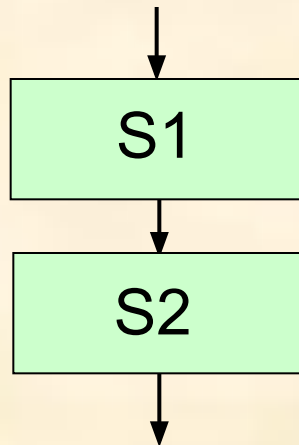
$\neg, \&, \vee, F, T$

# Основные управляющие структуры и инструкции (операторы) языка C++

## Управляющие структуры

= способы сочетания простых операторов  
= управление вычислительным процессом  
(порядком действий)

1. **Структура следования** =  
последовательность операторов



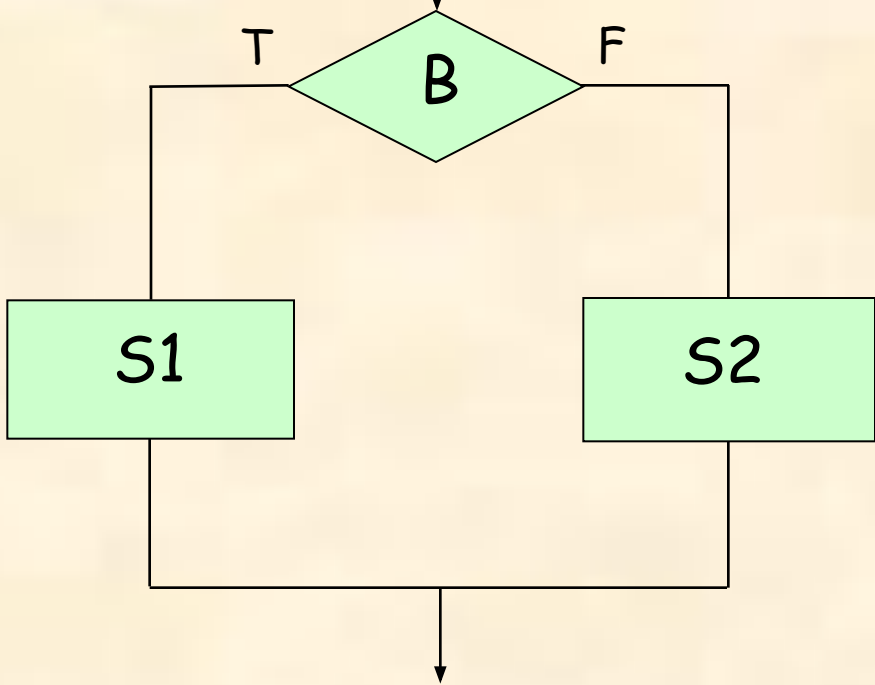
Последовательное выполнение  $S1; S2;$

Операторные скобки -  $\{ \dots \}$

$\{ S1; S2; \}$

Составной оператор

## 2. Структура ветвления = условный оператор = инструкция выбора



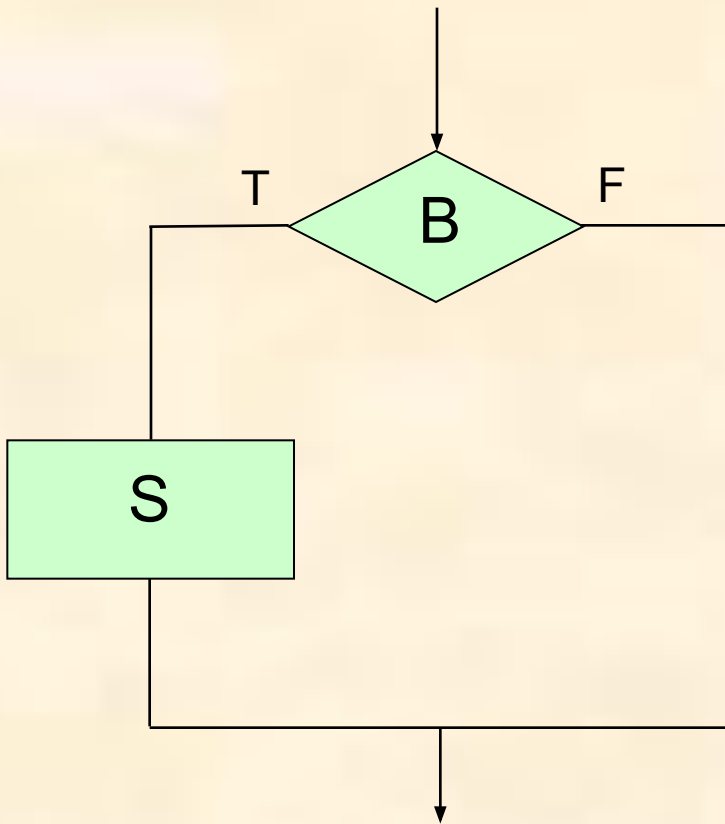
if (B) S1 else S2 =

$\left. \begin{array}{l} S1, \text{ при } B \\ S2, \text{ при } \text{not } B \end{array} \right\} =$

B - условие = булевское выражение  
S1 и S2 - инструкции

**Пример:**

`if (a > b) a = a - b; else b = b - a;`



Сокращенный условный оператор

**if** (B) S ≡

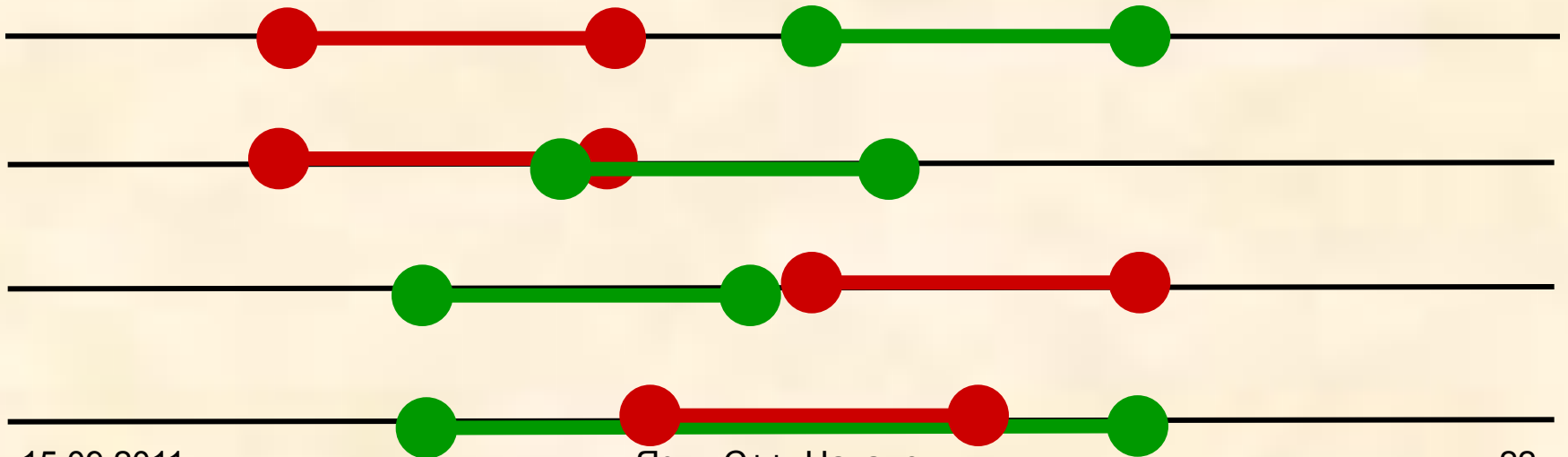
≡ **if** (B) S **else** ПУСТО

# Пример 1: проверка пересечения отрезков

**Дано:** два отрезка на прямой  $[a, b]$  и  $[c, d]$  с целочисленными координатами концов  $a, b, c, d \in \mathbb{Z}$  ( $a \leq b, c \leq d$ )

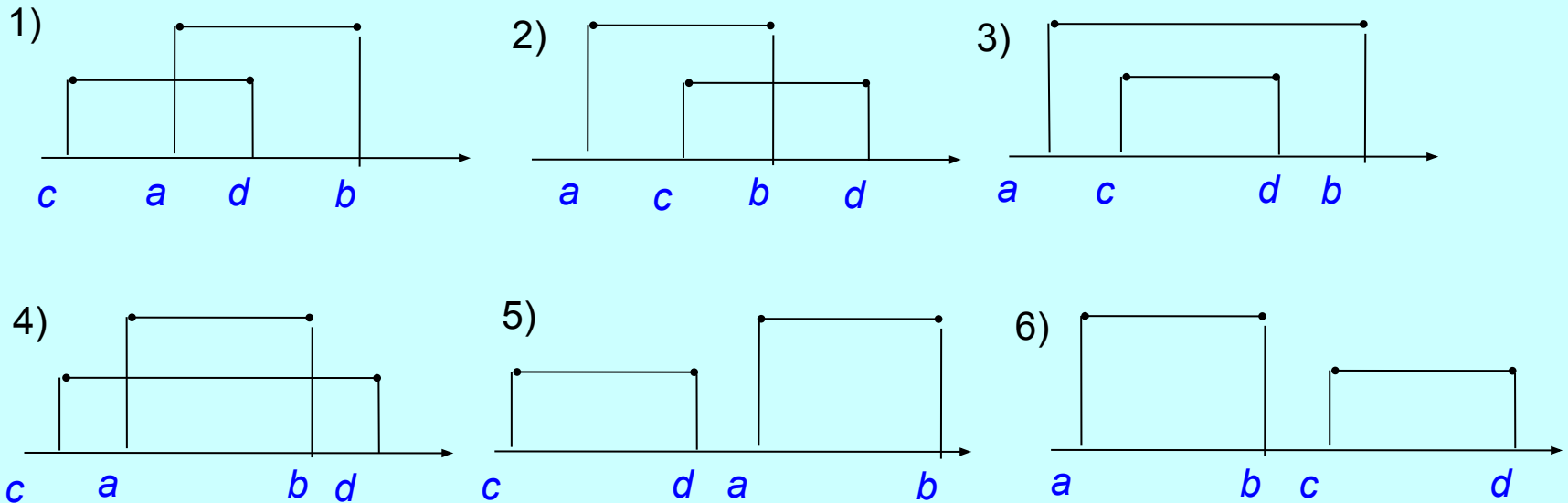
**Требуется:**

определить - пересекаются ли отрезки  $[a, b]$  и  $[c, d]$



# Условие пересечения

$$(b \geq c) \& (d \geq a)$$



В вариантах 1) – 4) справедливо  $(b \geq c) \& (d \geq a)$ ;

в варианте 5) –  $(b \geq c) \& (d < a)$ ;

в варианте 6) –  $(b < c) \& (d \geq a)$

# Инструкция выбора

```
if (( $b \geq c$ ) && ( $d \geq a$ )) отрезки пересекаются;  
else отрезки не пересекаются;
```

См. программу в файле [inters.cpp](#)

**Замечание** о корректности исходных данных



## Пример программы

```
// Пример 1.1 : пересечение отрезков
#include <iostream>
using namespace std ;
int main ( )
{   int a, b, c, d, w; // входные данные
    cout << "Введите координаты первого отрезка (левый и правый концы):\n" ;
    cin >> a >> b;
    cout << "Введите координаты второго отрезка (левый и правый концы):\n" ;
    cin >> c >> d;
    cout << "Введены отрезки: [" << a << ", " << b << "] [" << c << ", " << d << "]\n";
    if (( b >= c ) && ( d >= a )) cout << "Отрезки пересекаются\n" ;
    else cout << "Отрезки не пересекаются\n" ;
    cout << "End of the program LINE\n" ;
    return 0 ;
}
```

Демонстрация выполнения. Обсуждение.

См. программу в файле [intrsct.cpp](#)

**Замечание** о корректности исходных данных (проверка?)

```

// Пример 1.2 : пересечение отрезков (с проверкой корректности ввода)
#include <iostream>
using namespace std ;
int main ( )
{   int a, b, c, d, w; // входные данные
    cout << "Введите координаты первого отрезка (левый и правый концы):\n" ;
    cin >> a >> b;
    if (a > b) { w = a; a = b; b = w; };
    cout << "Введите координаты второго отрезка (левый и правый концы):\n" ;
    cin >> c >> d;
    if (c > d) { w = c; c = d; d = w; };
    cout << "Введены отрезки: [" << a << ", " << b << "] [" << c << ", " << d << "]\n";
    if (( b >= c ) && ( d >= a )) cout << "Отрезки пересекаются\n" ;
    else cout << "Отрезки не пересекаются\n" ;
    cout << "End of the program LINE\n" ;
    return 0 ;
}

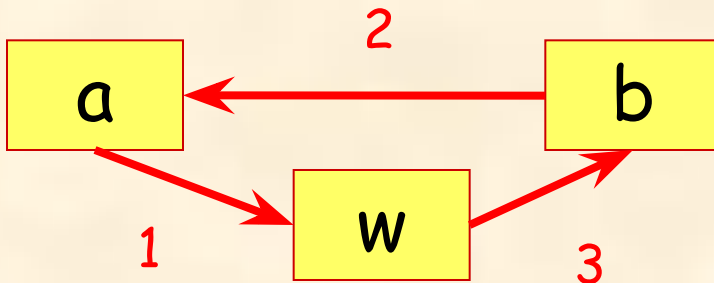
```

## Обмен значений

```
if (a > b) { w = a; a = b; b = w; };
```



```
if (a > b) a ⇔ b;
```



// Пример 1.3 : пересечение отрезков (с вычислением булевой переменной)

```
#include <iostream>
```

```
using namespace std ;
```

```
int main ( )
```

```
{ int a, b, c, d, w; // входные данные
```

```
bool p;
```

```
cout << "Введите координаты первого отрезка (левый и правый концы):\n" ;
```

```
cin >> a >> b;
```

```
if (a > b) { w = a; a = b; b = w;};
```

```
cout << "Введите координаты второго отрезка (левый и правый концы):\n" ;
```

```
cin >> c >> d;
```

```
if (c > d) {w = c; c = d; d = w;};
```

```
cout << "Введены отрезки: [" << a << "," << b << "] [" << c << "," << d << "]\n";
```

```
p = ( b >= c ) && ( d >= a );
```

```
cout << "Отрезки " ;
```

```
if (!p ) cout << "НЕ" ;
```

```
cout << " пересекаются\n" ;
```

```
cout << "End of the program LINE INTERSECTION\n" ;
```

```
return 0 ; }
```

## ВарианТЫ

```
p = ( b >= c ) && ( d >= a );  
// p - отрезки пересекаются  
if (!p ) cout << "HE" ;
```

$!p \Rightarrow !( ( b \geq c ) \&\& ( d \geq a ) ) \Rightarrow$   
 $\Rightarrow !( b \geq c ) \|\| !( d \geq a ) \Rightarrow ( b < c ) \|\| ( d < a )$

```
q = ( b < c ) \|\| ( d < a );  
// q - отрезки не пересекаются  
if (q) cout << "HE" ;
```

Проще было бы  
начать именно с  
условия  
«непересечения»!

Закон Де Моргана:  
 $\neg ( a \& b ) = ( \neg a ) \vee ( \neg b ) .$

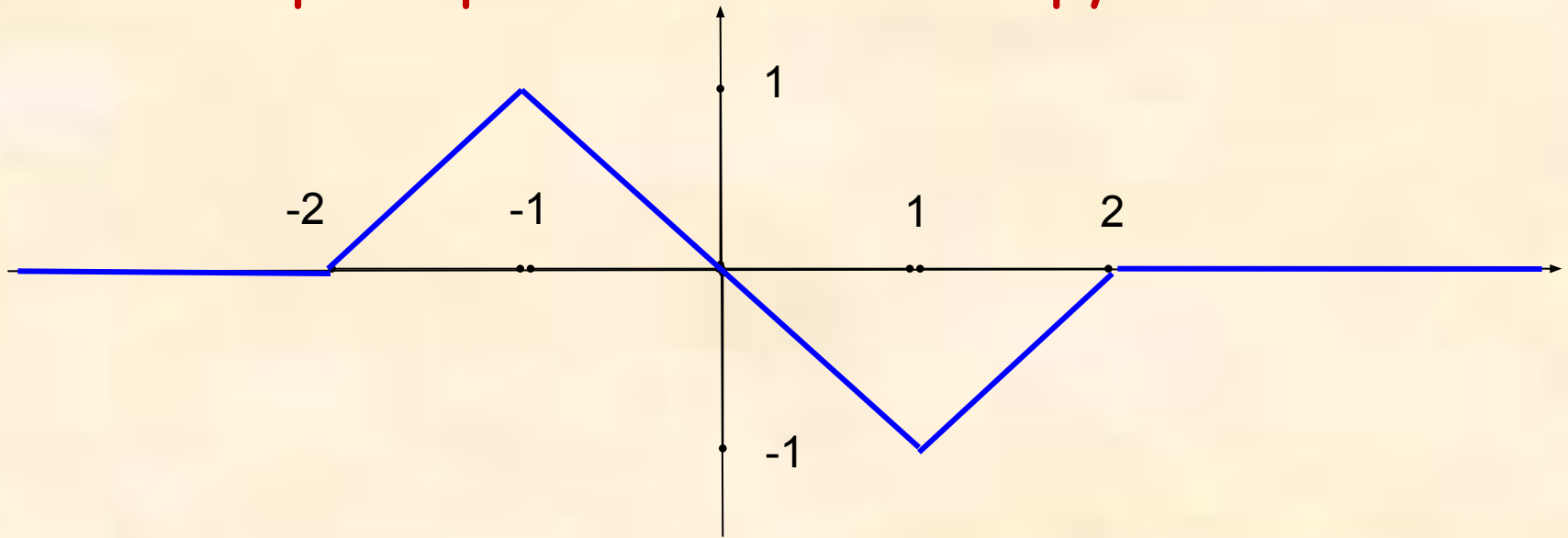
Конец примера 1

## Указание

Следующие примеры на лекции не рассматривались (но были на практическом занятии в некоторых группах).

Студенты должны разобрать эти примеры **самостоятельно**.

## Пример 2. Вычисление функции



Функция  $f(x)$  задана графиком (см.рис.).

Дано:  $x$  - вещественное число

Требуется: вычислить значение функции  $y=f(x)$

## Анализ задачи

Запишем аналитическое представление функции:

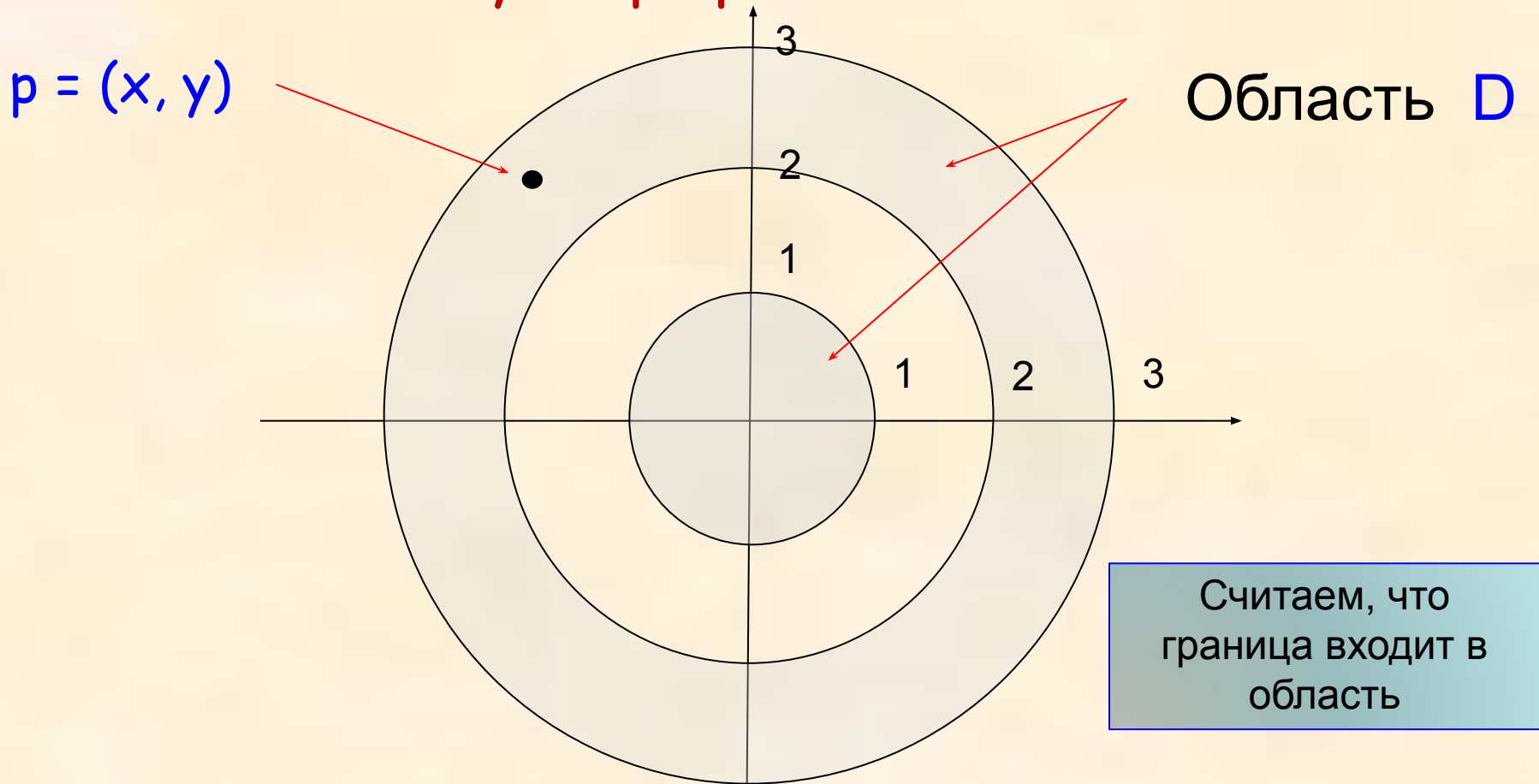
$$y = \begin{cases} 0, & \text{при } x \leq -2 \\ x + 2, & \text{при } -2 < x \leq -1 \\ -x, & \text{при } -1 < x \leq 1 \\ x - 2, & \text{при } 1 < x \leq 2 \\ 0, & \text{при } x > 2 \end{cases}$$



# Пример программы

```
// пример 2 с инструкцией выбора (ветвления)
#include <iostream>
using namespace std;
int main ( )
{ float x, y;
  cout << "Enter argument x : " ;
  cin >> x ;
  cout << "Argument x = " << x << endl;
  y = 0;
  if ((-2 < x) && (x <= -1)) y = x + 2;
  else if ((-1 < x) && (x <= 1)) y = - x;
  else if ((1 < x) && (x <= 2)) y = x - 2;
  cout << "Value function f (" << x << ") = " << y << endl;
  return 0;
}
```

# Пример 3. Попадание точки на плоскости в заданную графически область



Дано: точка  $p = (x, y)$

Ответ:

"точка лежит/не лежит в  $D$ "

## Анализ задачи

Уравнение окружности радиуса  $r$ :  $x^2 + y^2 = r^2$

Круг радиуса  $r_1$ :  $x^2 + y^2 \leq r_1^2$

Кольцо с радиусами  $r_2$  и  $r_3$  ( $r_2 \leq r_3$ ):

$r_2^2 \leq x^2 + y^2 \leq r_3^2$  или в другой форме  
( $x^2 + y^2 \geq r_2^2$ ) & ( $x^2 + y^2 \leq r_3^2$ )

---

В нашей задаче: точка  $p = (x, y)$ , область  $D$ ,

$p \in D \Leftrightarrow (p \in \text{Круг}) \vee (p \in \text{Кольцо})$

$r_1 = 1, r_2 = 2, r_3 = 3$

$(x^2 + y^2 \leq 1) \vee (x^2 + y^2 \geq 4) \& (x^2 + y^2 \leq 9)$

// пример 3 с инструкцией выбора (ветвления) - вариант 3.1

```
#include <iostream>
```

```
using namespace std ;
```

```
int main ( )
```

```
{ float x, y, r2;
```

```
bool b = false;
```

```
cout << "Enter x and y for point p=(x y): ";
```

```
cin >> x >> y;
```

```
cout << "Point p = (" << x << " , " << y << " ) " << endl;
```

```
r2 = x*x + y*y;
```

```
if (r2 <= 1) b = true;
```

```
else if (r2 >= 4) if (r2 <= 9) b = true;
```

```
if (!b) cout << "NOT ";
```

```
cout << "in region !" << endl;
```

```
return 0;
```

```
}
```

// пример 3 с инструкцией выбора (ветвления) - вариант 3.2

```
#include <iostream>
```

```
using namespace std ;
```

```
int main ( )
```

```
{ float x, y, r2;
```

```
bool b = false;
```

```
cout << "Enter x and y for point p=(x y): ";
```

```
cin >> x >> y;
```

```
cout << "Point p = (" << x << " , " << y << " ) " << endl;
```

```
r2 = x*x + y*y;
```

```
if ( (r2 <= 1) || (r2 >= 4) && (r2 <= 9) ) b = true;
```

```
if (!b) cout << "NOT";
```

```
cout << "in region !" << endl;
```

```
return 0;
```

```
}
```

// пример 3 с инструкцией выбора (ветвления) - вариант 3.3

```
#include <iostream>
```

```
using namespace std ;
```

```
int main ( )
```

```
{ float x, y, r2;
```

```
bool b = false;
```

```
cout << "Enter x and y for point p=(x y): ";
```

```
cin >> x >> y;
```

```
cout << "Point p = (" << x << " , " << y << " ) " << endl;
```

```
r2 = x*x + y*y;
```

```
b = (r2 <= 1) || (r2 >= 4) && (r2 <= 9);
```

```
if (!b) cout << "NOT";
```

```
cout << "in region !" << endl;
```

```
return 0;
```

```
}
```

Проанализировать отличие  
вариантов 1, 2 и 3.

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ