

A decorative graphic on the left side of the slide, consisting of a network of light blue lines and circles that resemble a circuit board or a data network. The lines are of varying thickness and connect to small circles, creating a complex, branching structure.

ENUM

# ОСНОВНЫЕ МОМЕНТЫ ИСПОЛЬЗОВАНИЯ ПЕРЕЧИСЛЕНИЙ

1. Использование перечислений вместо констант облегчает чтение программы.
2. Идентификаторы в `enum` должны быть уникальными, но отдельные константы перечисления могут иметь одинаковые значения.
3. Набор идентификаторов перечисляемого типа – собственный уникальный тип, отличающийся от других целочисленных типов.
4. Перечисляемые константы могут определяться и инициализироваться произвольными целочисленными константами, а также константными выражениями:

```
enum ages {Andrey = 47, Ira ← 48  
          Petya = 56, Yura = Petya + 7};
```

# ОСНОВНЫЕ МОМЕНТЫ ИСПОЛЬЗОВАНИЯ ПЕРЕЧИСЛЕНИЙ

5 Каждое перечисление является отдельным типом. Типом элемента перечисления является само перечисление.

ТИП Keyboard

```
enum Keyword {ASM, AUTO, BREAK};
```

6. Перечисляемая константа может быть объявлена анонимно, то есть без имени типа

```
enum {FALSE, TRUE};
```

```
enum {lazy, hazy, crazy} why;
```

Первое объявление – распространенный способ объявления мнемонической целочисленных констант.

Второе объявление объявляет переменную `why` с допустимыми значениями `lazy`, `hazy`, `crazy`.

# ОСНОВНЫЕ МОМЕНТЫ ИСПОЛЬЗОВАНИЯ ПЕРЕЧИСЛЕНИЙ

7. Перечисления могут неявно преобразовываться в обычные целочисленные типы, но не наоборот

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

void main()
{
    enum boolean { FALSE, TRUE } q;
    enum signal { off, on } a = on; //a инициализируется в on
    enum answer { no, yes, maybe = -1 } b;

    int i, j = true; // верно true преобразуется в 1
    a = off; // верно
    i = a; //верно i становится 0
    q = a; // неверно, два разных типа
    q = (boolean)a // верно, явное преобразование приведением
}
}
```