

# Перечисление (enum)

**Перечисление** (или ещё «**перечисляемый тип**») — это тип данных, где любое значение (или ещё «**перечислитель**») определяется как [символьная константа](#).

```
// Объявляем новое перечисление Colors
enum Colors
{
// Ниже находятся перечислители
// Это все возможные значения этого типа данных
// Каждый перечислитель разделяется запятой (НЕ точкой с запятой)
COLOR_RED,
COLOR_BROWN,
COLOR_GRAY,
COLOR_WHITE,
COLOR_PINK,
COLOR_ORANGE,
COLOR_BLUE,
COLOR_PURPLE, //
}; // Однако сам enum должен заканчиваться точкой с запятой

// Определяем несколько переменных перечисляемого типа Colors
Colors paint = COLOR_RED;
Colors house(COLOR_GRAY);
```

# Значения перечислителей

```
enum Colors
{
COLOR_YELLOW, // присваивается 0
COLOR_WHITE, // присваивается 1
COLOR_ORANGE, // присваивается 2
COLOR_GREEN, // присваивается 3
COLOR_RED, // присваивается 4
COLOR_GRAY, // присваивается 5
COLOR_PURPLE, // присваивается 6
COLOR_BROWN // присваивается 7
};
```

```
int main()
{
Colors paint(COLOR_RED);
std::cout << paint;
}
```

```
enum Animals
{
ANIMAL_PIG = -4,
ANIMAL_LION, // присваивается -3
ANIMAL_CAT, // присваивается -2
ANIMAL_HORSE = 6,
ANIMAL_ZEBRA = 6, // имеет то же значение, что
и ANIMAL_HORSE
ANIMAL_COW // присваивается 7
};
```

# Обработка перечислений

```
enum Animals
{
ANIMAL_PIG = -4,
ANIMAL_LION, // присваивается -3
ANIMAL_CAT, // присваивается -2
ANIMAL_HORSE = 6,
ANIMAL_ZEBRA = 6, // имеет то же значение, что и ANIMAL_HORSE
ANIMAL_COW // присваивается 7
};

int main()
{
int mypet = ANIMAL_PIG;
cout << ANIMAL_HORSE; // конвертируется в int, а затем выводится на экран
}
```

*Animals animal = 7; // приведёт к ошибке компиляции*

*cin >> color; // приведёт к ошибке компиляции*

*Animals animal = COLOR\_BLUE; // приведёт к ошибке компиляции*

# Вывод перечислений

```
enum Colors
{
    COLOR_PURPLE, // присваивается 0
    COLOR_GRAY, // присваивается 1
    COLOR_BLUE, // присваивается 2
    COLOR_GREEN // присваивается 3
};

void printColor(Colors color)
{
    if (color == COLOR_PURPLE)
        cout << "Purple";
    else if (color == COLOR_GRAY)
        cout << "Gray";
    else if (color == COLOR_BLUE)
        cout << "Blue";
    else if (color == COLOR_GREEN)
        cout << "Green";
    else
        cout << "Who knows!";
}
```

# Использование перечислений

```
int readFileContents()
{
    if (!openFile())
        return -1;
    if (!parseFile())
        return -2;
    if (!readFile())
        return -3;
    return 0; // если всё прошло успешно
}
```

```
if (readFileContents() == SUCCESS)
{
    // Делаем что-нибудь
}
else
{
    // Выводим сообщение об ошибке
}
```

```
enum ParseResult
{
    SUCCESS = 0,
    ERROR_OPENING_FILE = -1,
    ERROR_PARSING_FILE = -2,
    ERROR_READING_FILE = -3
};

ParseResult readFileContents()
{
    if (!openFile())
        return ERROR_OPENING_FILE;
    if (!parseFile())
        return ERROR_PARSING_FILE;
    if (!readfile())
        return ERROR_READING_FILE;

    return SUCCESS; // если всё прошло
успешно
}
```

# Использование перечислений

```
enum ItemType
{
    ITEMTYPE_GUN,
    ITEMTYPE_ARBALET,
    ITEMTYPE_SWORD
};

std::string getItemName(ItemType itemType)
{
    if (itemType == ITEMTYPE_GUN)
        return std::string("Gun");
    if (itemType == ITEMTYPE_ARBALET)
        return std::string("Arbalet");
    if (itemType == ITEMTYPE_SWORD)
        return std::string("Sword");
}

int main()
{
    // ItemType - это перечисляемый тип, который мы объявили выше
    // itemType (с маленькой i) - это имя переменной, которую мы определили (типа ItemType)
    // ITEMTYPE_GUN - это значение перечислителя, которое мы присвоили переменной itemType
    ItemType itemType(ITEMTYPE_GUN);

    std::cout << "You are carrying a " << getItemName(itemType) << "\n";
}
```

# Использование перечислений

```
enum SortType
{
    SORTTYPE_FORWARD,
    SORTTYPE_BACKWARDS
};

void sortData(SortType type)
{
    if (type == SORTTYPE_FORWARD)
        // Сортировка данных в одном порядке
    else if (type == SORTTYPE_BACKWARDS)
        // Сортировка данных в обратном порядке
}
```