?: ??????????

Переменные бывают локальные (в пределах функции) и глобальные. Глобальные определяются до функции main вне остальных функций.

Переменные можно просто объявлять и дополнительно инициализировать начальное значение. Размеры типов зависят от платформы и компилятора! Вывод размера:

```
sizeof(имя_типа);
```

Квалификаторы типов (модификаторы)

Они не создают новые типы, но изменяют их свойства.
const Значение нельзя изменить const int x = 5;
volatile Значение может измениться внешне (прерывания, hardware) volatile int status;
restrict Указатель — единственный способ доступа к данным (для оптимизации) int *restrict ptr;
signed Тип со знаком (по умолчанию для char не определено) signed char c;
unsigned Тип без знака (только положительные числа) unsigned int u;

Псевдонимы типов (typedef)

Позволяет дать новое имя существующему типу.

```
typedef int my_int;           // my_int — это псевдоним для int
typedef unsigned long ulong;  // ulong — псевдоним для unsigned long

my_int x = 10;                // То же, что int x = 10;
ulong y = 100;                // То же, что unsigned long y = 100;
```

Часто используется для упрощения сложных типов:

```
typedef struct {
    char name[50];
    int age;
} Person;                     // Теперь можно писать Person p; вместо struct Person p;
```

Базовые типы:

void	Нет значений Означает "отсутствие типа"	Нет размера	

integer (int)	Целые числа, -2,147,483,648 … 2,147,483,647 Зависит от разрядности ОС!	4	`int var;` `int var=7;`
unsigned int	Беззнаковое целое 0 … 4,294,967,295	4	
short	Числа со знаком -32,768 … 32,767	2	
unsigned short	Беззнаковое короткое целое 0 … 65,535	2	
long		4 или 8 Зависит от платформы	
unsigned long			
long long	Очень длинное целое	8 байт	
unsigned long long		8	
float	Дробные с точностью до 7 знака	4	`float var_float = 12.45;`
double	Дробные с точностью 15 знаков	8	
long double		10, 12 или 16 Зависит от платформы	
bool	Логическое		`bool vr = true;`
char	Символ Желательно указывать явно signed / unsigned	1	`char n = 'g';`
unsigned char	Беззнаковый символ 0 … 255	1	

Массивы

Хранят несколько элементов одного типа подряд в памяти. Из базовых.

```
int numbers[10];          // Массив из 10 целых чисел
char name[50];            // Массив из 50 символов
```

```
double matrix[3][3];       // Двумерный массив (матрица 3x3)
```

Указатели (pointer)

Хранят адрес в памяти, где находится значение. Для обращения по адресам необходимы:

1. Возможность находить адрес, по которому в памяти располагается та или иная переменная;
2. Место, где этот адрес можно было бы сохранить для дальнейшей работы;
3. Возможность обращаться к ячейке памяти по ее адресу.

Операция получения адреса переменной & (амперсанд). Хранятся адреса переменных тоже в переменных, в специальном типе,
переменные указатели. Количество указательных типов, равно количеству типов данных.
Объявление переменной-указателя

```
int *ptr, i;               // Указатель на int и обычный int
char *str;                 // Указатель на char (строка)
void *generic;             // Универсальный указатель (на любой тип)
```

Для изменения / получения данных из адреса оператор разыменовывания * но в операциях с переменной:

```
char i = 0, *p;
p = &i;
*p = 2;
i = i + 2;
printf("%d %d", *p, i);
```

Но если попытаться напрямую изменить адрес, например p=p+2; то будет ошибка (такого адреса нет).

В ячейке памяти хранится адрес первой ячейки памяти. Тип переменной указывается для понимания, сколько байт нужно считывать.

Перечисления (enum)

Определяют набор именованных целочисленных констант.

```
enum Color {
    RED,    // 0
    GREEN,  // 1
```

```
    BLUE    // 2
};

enum Color favorite = GREEN;
```

Структуры

Объединяют разные типы в один.

```
struct Person {
    char name[50];
    int age;
    float height;
};

struct Person person1;
```

Объединения (union)

Как структура, но все поля используют одну и ту же область памяти (экономят память).

```
union Data {
    int i;
    float f;
    char str[20];
};

union Data data;           // data.i, data.f, data.str занимают одну память
```

Преобразование типов.

Бывают явные (делает программист) и неявные (компилятор). Неявные делятся на:

1. При присваивании результата вычисления переменной, тип результата к преобразуется к типу переменной. Округление происходит не по математическим законам, а путем отбрасывания значения.
2. При вычислении выражения операнды каждой операции преобразуются к какому-то одному типу (если они различаются), при этом результат операции будет того же типа. Правило «меньший» тип к «большему», для минимизации потерю информации. Операции преобразования сначала выполняются от самого глубокого до внешнего, одного приоритета слева направо.

```
float c = 5/6;
```

5 и 6 - целочисленные, поэтому результат целочисленный. И равен 0. Только после этого он преобразуется в float, но уже поздно. В с будет 0. А если сделать один операнд с плавающей точкой - второй преобразуется во float и дробная часть не потеряется.

```
float c = 5.0/6;
```

Пример приоретизации операций:

```
2 / 4 * 3,14
// результат 0, так как одноуровневые операции слева направо
```

Явное преобразование:

```
double result = (double)a / b;
```

Revision #4
Created 17 July 2026 16:07:02 by Admin
Updated 19 July 2026 08:47:24 by Admin