ПЕРЕМЕННЫЕ В ЯЗЫКЕ С

ПЕРЕМЕННЫЕ В ЯЗЫКЕ С .

Переменная *— это именованная область памяти, в которой хранятся данные определенного типа.*

У переменной есть имя и значение.

- Имя служит для обращения к области памяти, в которой хранится значение.
- Во время выполнения программы значение переменной можно изменять.

Перед использованием любая переменная должна быть описана.

Общий вид оператора описания переменных:

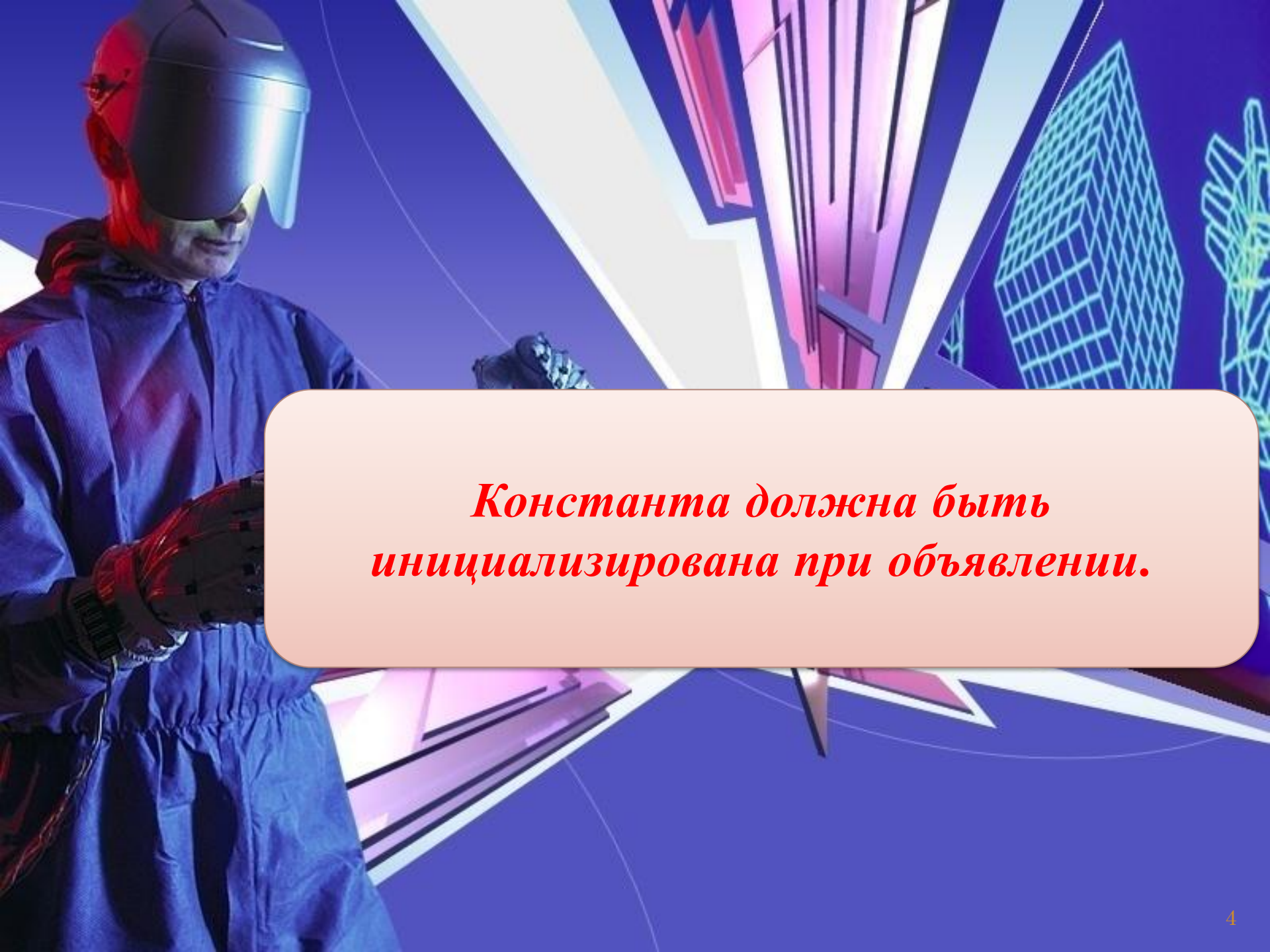
[класс памяти] [const] тип имя [инициализатор];

класс памяти
- может принимать одно из значений **auto**, **extern**, **static** и **register**
- указывает

модификатор const
- переменной изменять нельзя
- присваивает

инициализатор
- начальное значение
- Инициализатор можно записывать в двух формах
  - со знаком равенства: **= значение**,
  - в круглых скобках: **(значение)**

Правила описания переменных



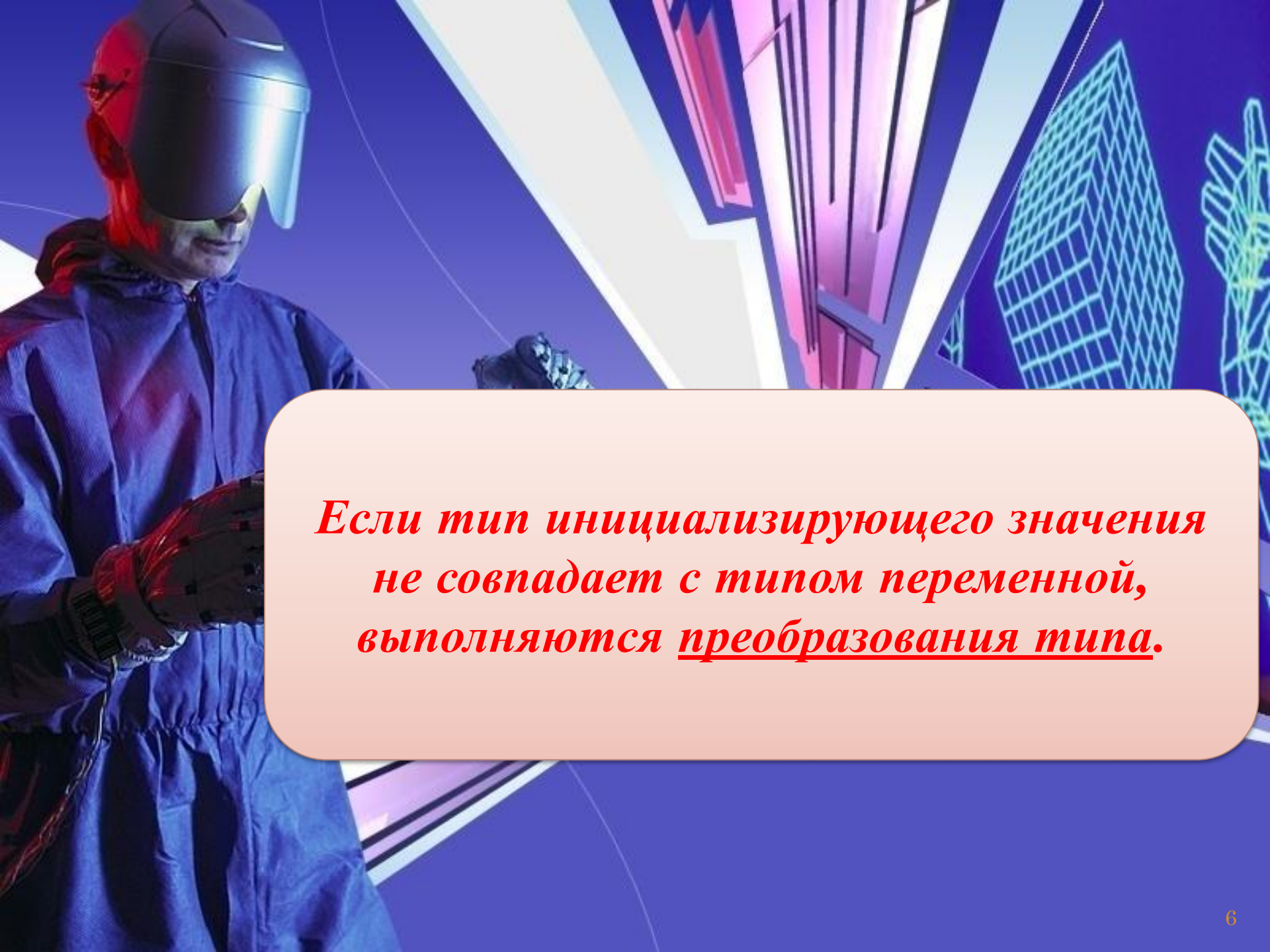
Константа должна быть инициализирована при объявлении.

В одном операторе можно описать несколько переменных одного типа, разделяя их запятыми

Примеры:

```
short int a = 1;          // целая переменная a

const char C = 'C';       // символьная константа C

char s, sf = 'f';         // инициализация относится  только к

sf.

char t (54);

float c = 0.22, x(3), sum;
```

Правила описания переменных



Если тип инициализирующего значения не совпадает с типом переменной, выполняются <u>преобразования типа</u>.

Любые операнды типа **char**, **unsigned char** или **short** преобразуются к типу **int**

- **char** расширяется нулем или знаком в зависимости от умолчания для **char**
- **unsigned char** расширяется нулем
- **signed char** расширяется знаком

Затем любые два операнда становятся либо **int**, либо **float**, **double** или **long double**

Правила преобразования типов

Если один из операндов имеет тип:	То другой преобразуется к типу :
long double	**long double**
double	**double**
float	**float**
unsigned long	**unsigned long**
long	**long**
unsigned	**unsigned**
Иначе оба операнда должны иметь тип **int**	
Тип результата тот же, что и тип участвующих в выражении операндов	

Правила преобразования типов

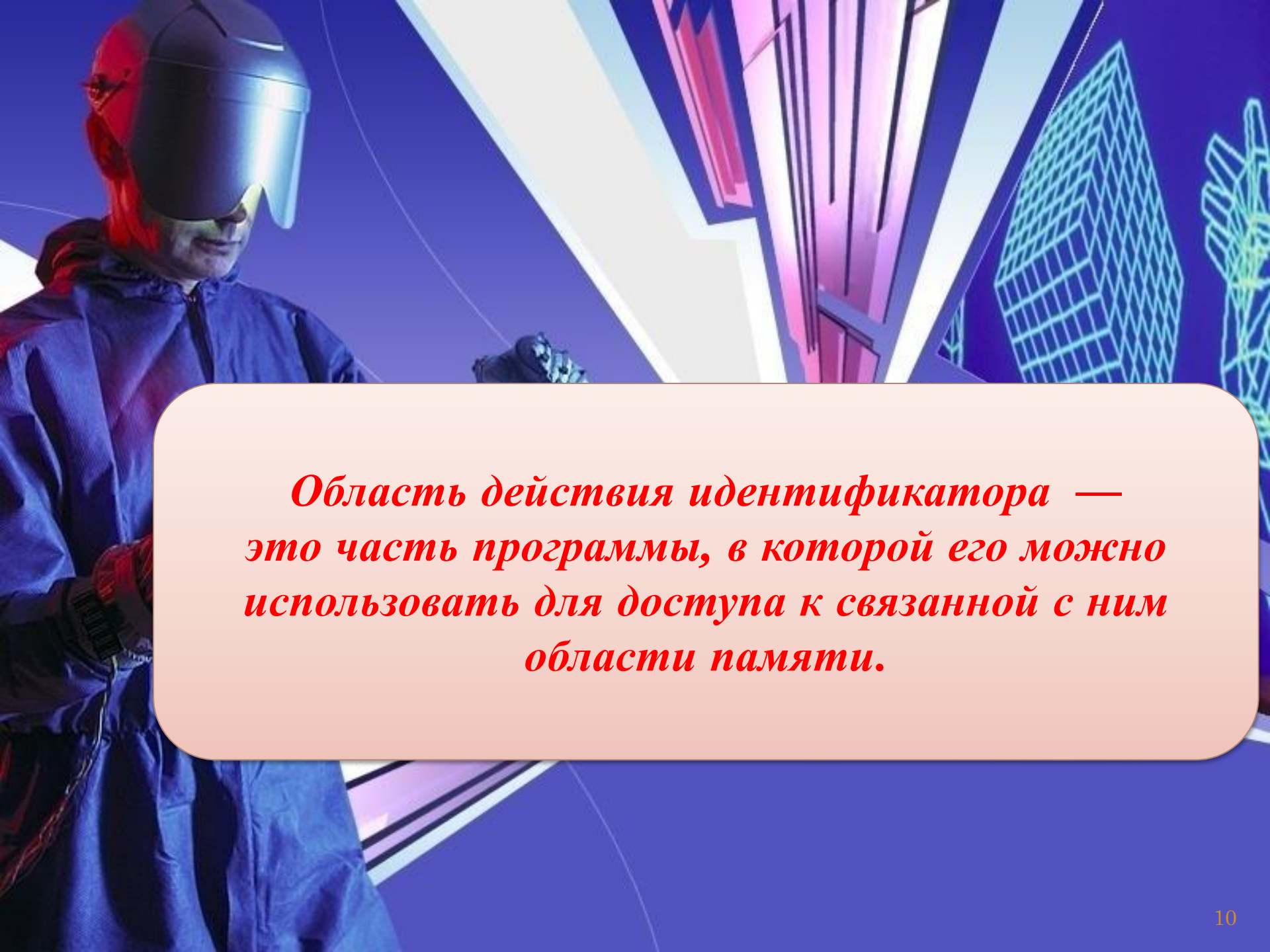
Описание переменной, кроме типа и класса памяти, явно или по умолчанию задает ее

Класс памяти и область действия

зависят не только от описания,

но и от места его размещения в тексте программы

Область действия идентификатора



Область действия идентификатора — это часть программы, в которой его можно использовать для доступа к связанной с ним области памяти.

*В зависимости от области действия переменная может быть **локальной** или **глобальной***

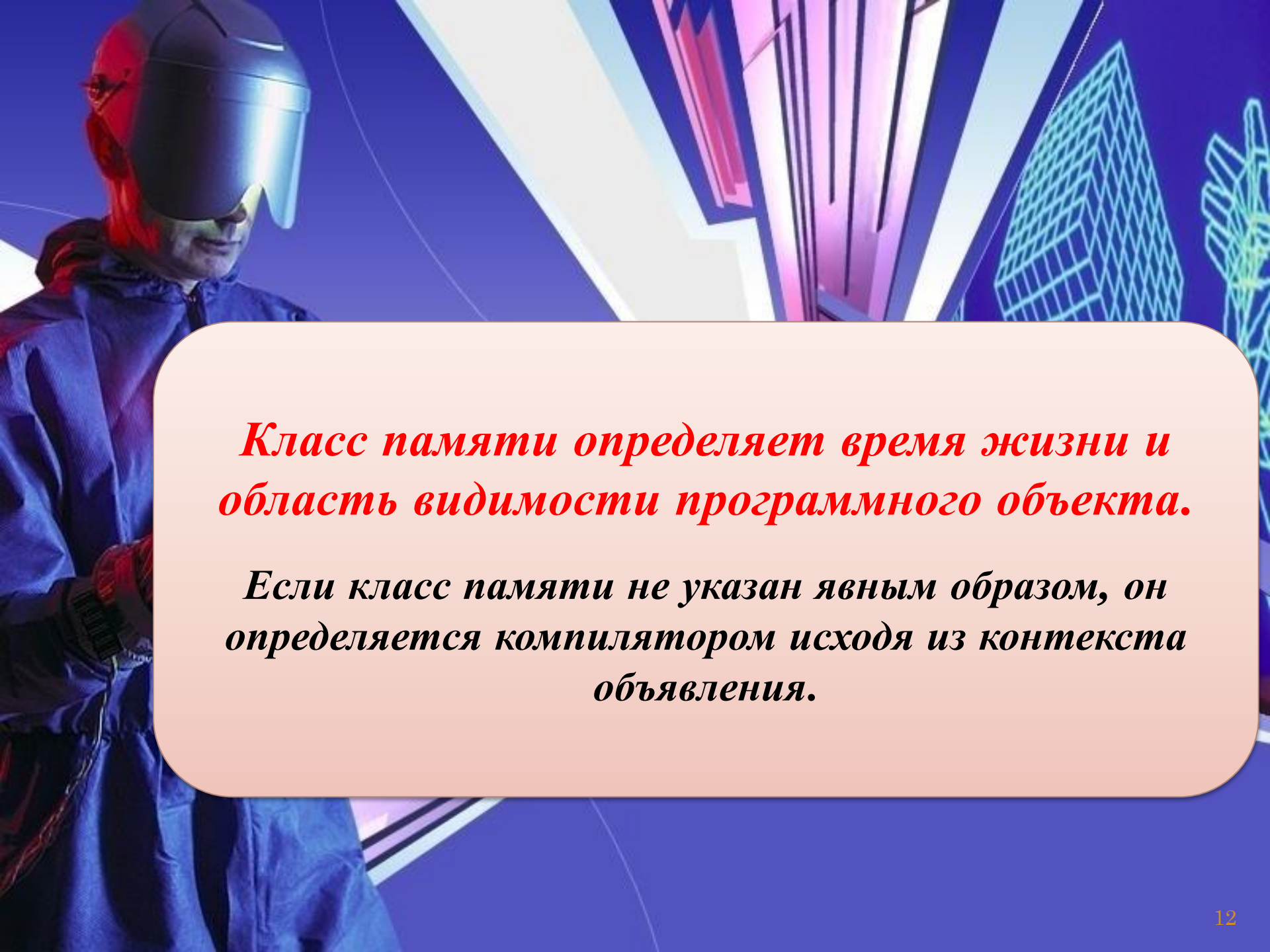
Если переменная определена внутри блока, она называется *локальной*

- область ее действия — от точки описания до конца блока, включая все вложенные блоки

Если переменная определена вне любого блока, она называется *глобальной*

- областью ее действия считается файл, в котором она определена, от точки описания до его конца

Область действия идентификатора



Класс памяти определяет время жизни и область видимости программного объекта.

Если класс памяти не указан явным образом, он определяется компилятором исходя из контекста объявления.

Время жизни может быть:

- постоянным - в течение выполнения программы
- временным - в течение выполнения блока

Класс памяти

Областью видимости идентификатора называется часть текста программы, из которой допустим обычный доступ к связанной с идентификатором областью памяти

- Чаще всего область видимости совпадает с областью действия

Исключением является ситуация, когда во *вложенном блоке* описана переменная с таким же именем

- *в этом случае внешняя переменная во вложенном блоке невидима, хотя он и входит в ее область действия*
- *тем не менее, к этой переменной, если она глобальная, можно обратиться, используя операцию доступа к области видимости* **::**

Класс памяти

Класс памяти

- Для задания *класса памяти* используются следующие спецификаторы:
  - **auto**
  - **extern**
  - **static**
  - **register**

Класс памяти

- **auto**
  - *автоматическая* переменная
  - память под нее выделяется в стеке и при необходимости инициализируется каждый раз при выполнении оператора, содержащего ее определение
  - освобождение памяти происходит при выходе из блока, в котором описана переменная.
  - время ее жизни — с момента описания до конца блока

Для глобальных переменных этот спецификатор не используется, а для локальных он принимается по умолчанию, поэтому задавать его явным образом большого смысла не имеет

- **extern**
  - означает, что переменная определяется в другом месте программы
  - (в другом файле или дальше по тексту)
  - используется для создания переменных, доступных во всех модулях программы, в которых они объявлены

*Если переменная в том же операторе инициализируется, спецификатор **extern** игнорируется*

Класс памяти

Класс памяти

- **static**
  - *статическая* переменная
  - время жизни — постоянное
  - инициализируется один раз при первом выполнении оператора, содержащего определение переменной
  - в зависимости от расположения оператора описания статические переменные могут быть глобальными и локальными

Глобальные статические переменные видны только в том модуле, в котором они описаны

- **register**
  - аналогично **auto**, но память выделяется по возможности в регистрах процессора
  - если такой возможности у компилятора нет, переменные обрабатываются как **auto**

Класс памяти

```
int a;          // 1 глобальная переменная a
int main(){
  int b;        // 2 локальная переменная b
  extern int x;  // 3 переменная x определена в другом месте
  static int c;  // 4 локальная статическая переменная c
  a = 1;         // 5 присваивание глобальной переменной
  int a;         // 6 локальная переменная a
  a = 2;         // 7 присваивание локальной переменной
  ::a = 3;         // 8 присваивание глобальной переменной
return 0;
}
int x = 4;         // 9 определение и инициализация x
```

Пример

В этом примере глобальная переменная **a** определена вне всех блоков.

память под нее выделяется в сегменте данных в начале работы программы

областью действия является вся программа

область видимости — вся программа, кроме строк 6-8

- так как в строке 6 определяется локальная переменная с тем же именем, область действия которой начинается с точки ее описания и заканчивается при выходе из блока

Пример

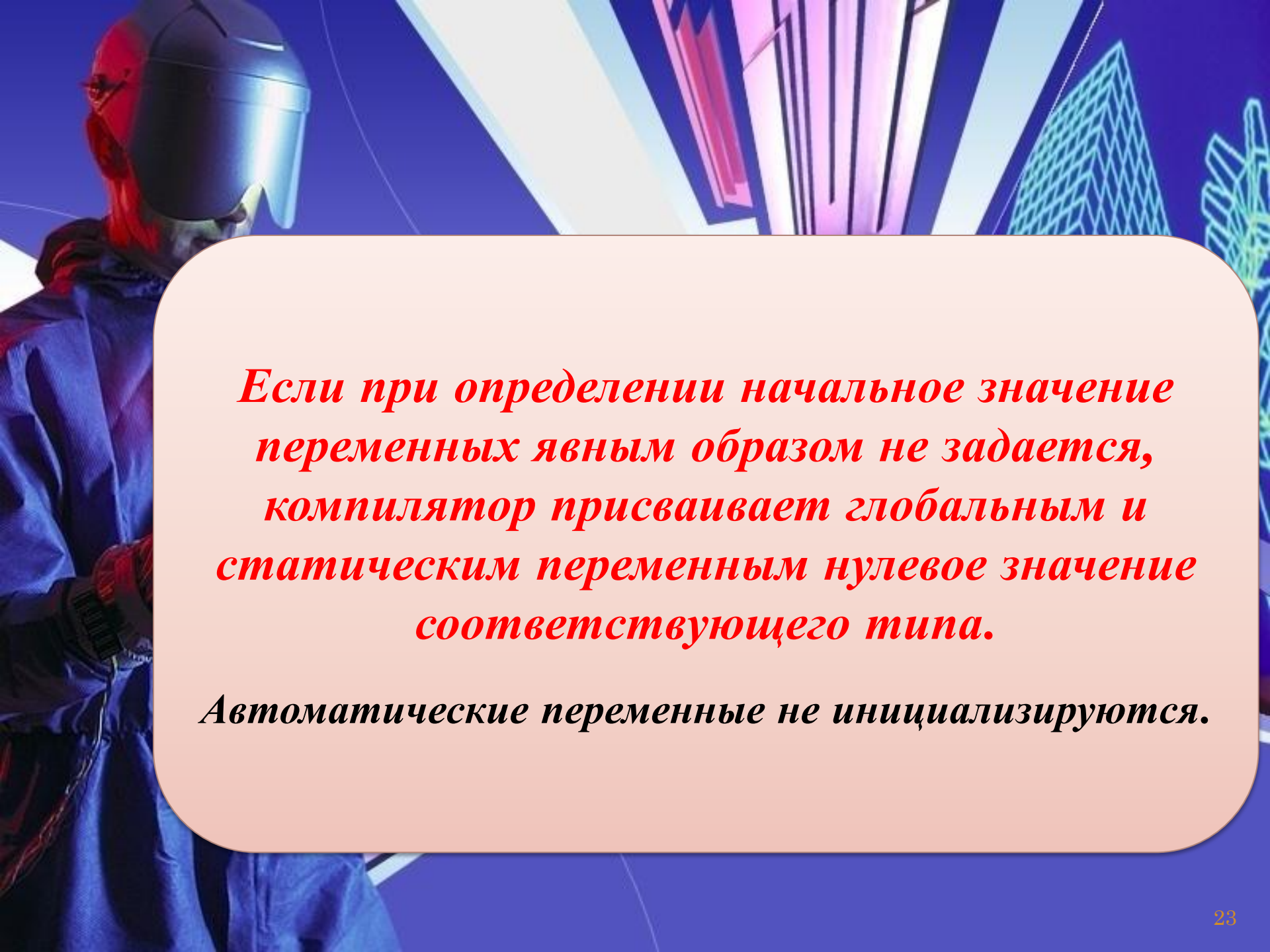
Переменные **b** и **c** — локальные.

область их видимости — блок

время жизни различно:

- память под **b** выделяется в стеке при входе в блок и освобождается при выходе из него
- переменная **c** располагается в сегменте данных и существует все время, пока работает программа

Пример



Если при определении начальное значение переменных явным образом не задается, компилятор присваивает глобальным и статическим переменным нулевое значение соответствующего типа.

Автоматические переменные не инициализируются.

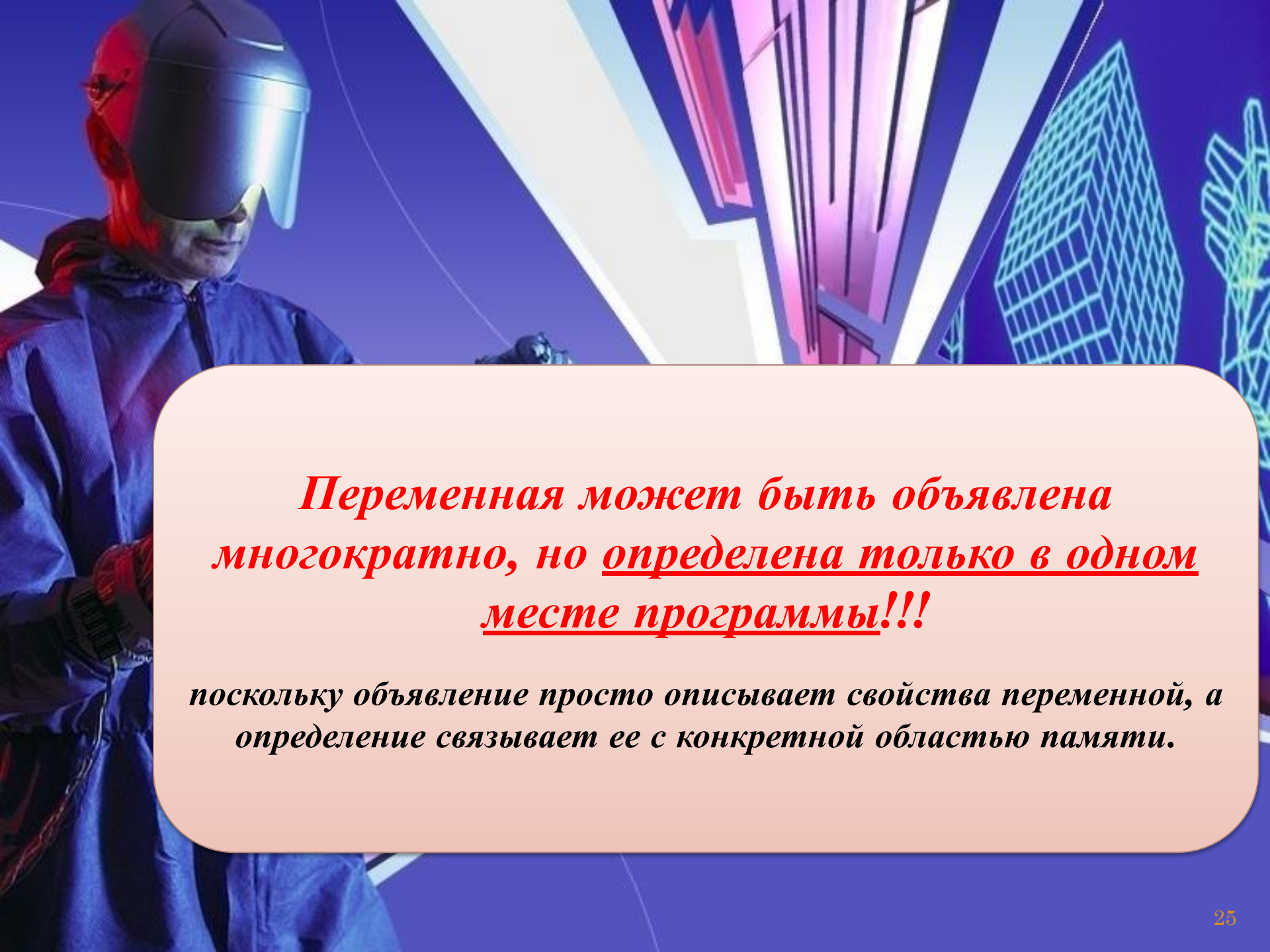
Описание переменной может выполняться в форме *объявления или определения*

- *объявление* информирует компилятор о типе переменной и классе памяти
- *определение содержит, кроме этого,* указание компилятору выделить память в соответствии с типом переменной

В **C** большинство объявлений являются одновременно и определениями

- в приведенном выше примере только описание 3 является объявлением, но не определением

Объявление и определение переменной



Переменная может быть объявлена многократно, но <u>определена только в одном месте программы</u>!!!

поскольку объявление просто описывает свойства переменной, а определение связывает ее с конкретной областью памяти.