



Тема 11. Препроцессор



Роль препроцессора при компиляции программы



ANSI-стандарт языка C описывает фазу, предшествующую переводу исходного кода программы в машинный код. Такая фаза выполняется препроцессором и включает:

- * "склеивание строк" - удаление пары \+перевод строки, получение лексем;*
- * обработку лексем - замену текста и макрорасширения;*
- * включение текста из других файлов в исходный файл;*
- * исключение определенных частей кода (условная трансляция).*



"Склеивание строк" и получение лексем

Препроцессор выбрасывает пару символов, состоящую из обратной наклонной черты (\) и перевода строки (\n). Разделительные символы (пробелы и знаки табуляции) роли при компиляции не играют.

Директивы препроцессора начинаются с символа # (этот символ должен стоять в начале строки, но перед ним также могут быть и пробелы) и заканчиваются концом строки.

Пример программы

```
prin\  
tf("Длинная ст\  
рока текста\n");
```



Длинная строка текста

Пример программы

```
if(   a < b   )  
    c = 5;  
a += c;
```



```
if( a < b ) c = 5; a += c;
```



Директивы препроцессора

Директива	Назначение
#	оператор расширения строк
##	оператор конкатенации лексем
#define	определение идентификатора или макроса
#undef	отмена определения
#if	оператор условной трансляции
#ifdef	оператор проверки определения
#ifndef	оператор проверки неопределенного имени
#else	блок else директивы if
#endif	завершение директивы if
#include	включить файл при компиляции
#error	выдача сообщения об ошибке
#line	задает номер следующей строки



Предопределенные имена препроцессора

Препроцессор имеет несколько заранее определенных идентификаторов и заменяет их специальной информацией. Эти идентификаторы нельзя повторно переопределять, к ним нельзя применять директиву **#undef**.

Стандартные имена

<code>__cplusplus</code>	Определено, если компилируется код C++.
<code>__DATE__</code>	Дата начала компиляции текущего файла.
<code>__FILE__</code>	Имя текущего файла.
<code>__FUNC__</code>	Имя текущей функции.
<code>__LINE__</code>	Номер текущей строки.
<code>__STDC__</code>	Определено, если применяется стандарт ANSI.
<code>__TIME__</code>	Время начала компиляции текущего файла.



Включение при компиляции кода из других файлов

```
Текст программы

#include "file1.h"
#include <file2.h>
...

Point points[100];
points[0].x = 200;
points[0].y = 120;
...

double area = PI*R*R;
```

```
Файл file1.h

const float PI = 3.14;
```

```
Файл file2.h

typedef struct
{
    int    x;
    int    y;
} Point;
```



Определение и отмена определения макроса

С помощью директивы препроцессора **#define** определяется макрос:

#define имя_макроса последовательность_лексем

Имя макроса должно отвечать требованиям к другим именам программы.

Последовательность лексем заканчивается концом строки (либо \ для продолжения).

При компиляции имя макроса заменяется на последовательность лексем.

Отменить определение макроса можно с помощью директивы **#undef**:

#undef имя_макроса

Текст программы 1

```
#define MAX 200
...
int data[MAX];

for(int i = 0; i < MAX; i++)
    data[i] = 0;
```

Текст программы 2

```
#define Red    0x0000FF
#define Green  0x00FF00
#define Blue   0xFF0000
...
int color = Red;
```

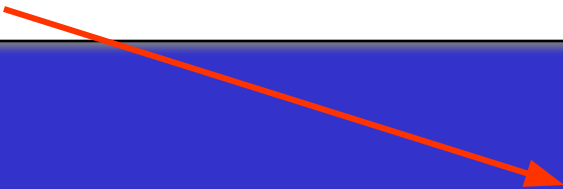


Макросы с параметрами

Макросы могут быть опеределены с аргументами, вследствие чего замещающий текст будет варьироваться в зависимости от задаваемых параметров.

Текст программы

```
#define P2(var) var*var
#define P3(var) var*var*var
...
double x, y, r, v;
...
r = sqrt(P2(x)+P2(y));
v = P3(r);
```



```
r = sqrt(x*x+y*y);
v = r*r*r;
```




Пример использования макроса

Текст программы 1

```
#define square(a, b) (a*b)
...
int s = square(3+1, 5+1);
```

s = 9

Текст программы 2

```
#define square(a, b) ((a)*(b))
...
int s = square(3+1, 5+1);
```

s = 24



Условная трансляция

*Директивы условной трансляции (**#if**, **#ifdef**, **#ifndef**, **#else**, **#endif**) позволяют выборочно включать в текст программы некоторые фрагменты в зависимости от значения заданных условий.*

*Директива **#if** начинает блок условной трансляции, который компилируется при выполнении заданного в директиве условия (константное целое выражение).*

*Директива **#ifdef** начинает блок условной трансляции, который компилируется, если заданное в директиве имя определено.*

*Директива **#ifndef** начинает блок условной трансляции, который компилируется, если заданное в директиве имя не определено.*

*Директива **#else** начинает блок условной трансляции, который компилируется при невыполнении заданного в директиве **#if** условия.*

*Директива **#endif** завершает блок условной трансляции.*



Пример условной трансляции

Текст программы

```
#define DEBUG
#define TRACE

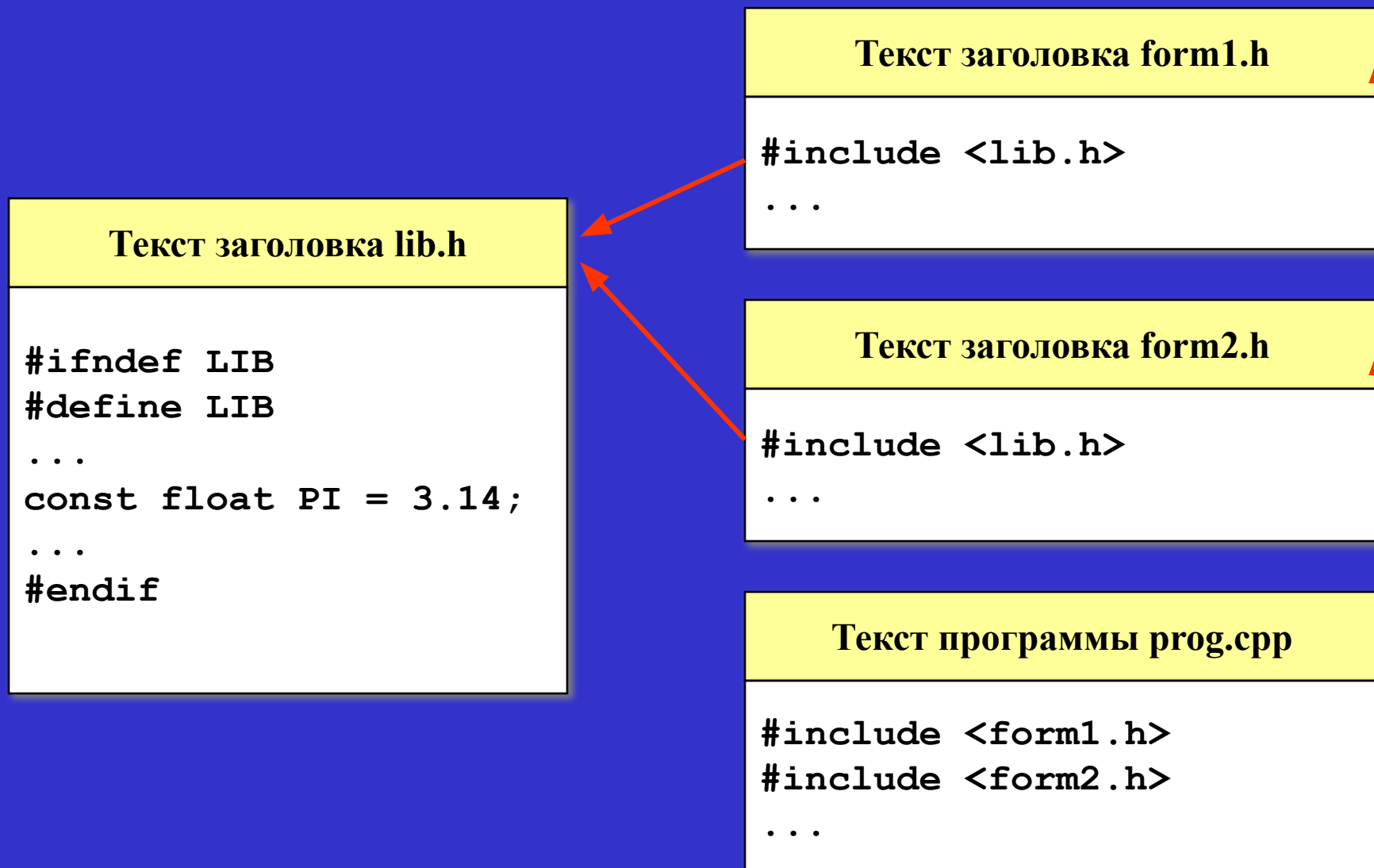
...

long password;

#ifdef DEBUG
#ifdef TRACE
    printf("Точка 1");
#endif
    password = 1;
#else
    GetPassword(password);
#endif
```



Пример условной трансляции





Расширение символьных строк

Оператор расширения символьных строк в макросах # позволяет преобразовать передаваемый макросу аргумент в символьную строку.

Пример программы

```
#define message(text) \  
    printf(#text);
```

...

```
message (Информация);
```

...

```
message ("Информация");
```

Информация

"Информация"



Конкатенация лексем

С помощью оператора конкатенации лексем ## отдельные лексемы "склеиваются" в одну. Оператор ## и все находящиеся между лексемами пробелы удаляются препроцессором.

Пример программы

```
#define message(var, num) printf("%d", var##num);  
...  
int code1 = 200;  
int code2 = 210;  
int code3 = 244;  
...  
message(code, 2);
```



Нумерация строк и сообщение об ошибке

С помощью директивы **#line** можно назначить номер строки внутри файла:

#line номер_строки [имя_файла].

Директива **#error** указывает на необходимость прекращения компиляции и вывода сообщения об ошибке:

#error текст_сообщения.

Пример программы

```
#line 100

#ifdef PARAMETER_X

#error Ошибка компиляции, не задан параметр X!

#endif;
```



Сравнение макросов и функций

