Препроцессор

- Макропроцессор, осуществляющий текстовую обработку на основе команд (директив), вставленных непосредственно в текст (нормальные алгорифмы Маркова)

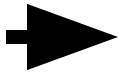
- Под программой на языке С обычно понимается программой на языке препроцессора языка С, из которой уже получается программа на С.

Препроцессор

Программа препроцессора

```
#define SMALL
#ifdef SMALL
#define N 10
#define number short int
#else
#define N 10000
#define number long int
#end if
#define reverse(k) N - k
number A[N];

void main()
{
    for (int i = 0; i<N; i++)
        A[i] = reverse(i) * reverse(i);
}
```



Программа на C

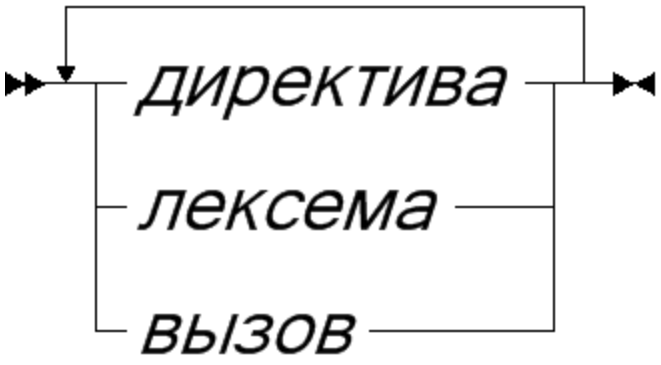
```
short int A[10];

void main()
{
    for (int i = 0; i<10; i++)
        A[i] = 10 - i * 10 - i;
}
```

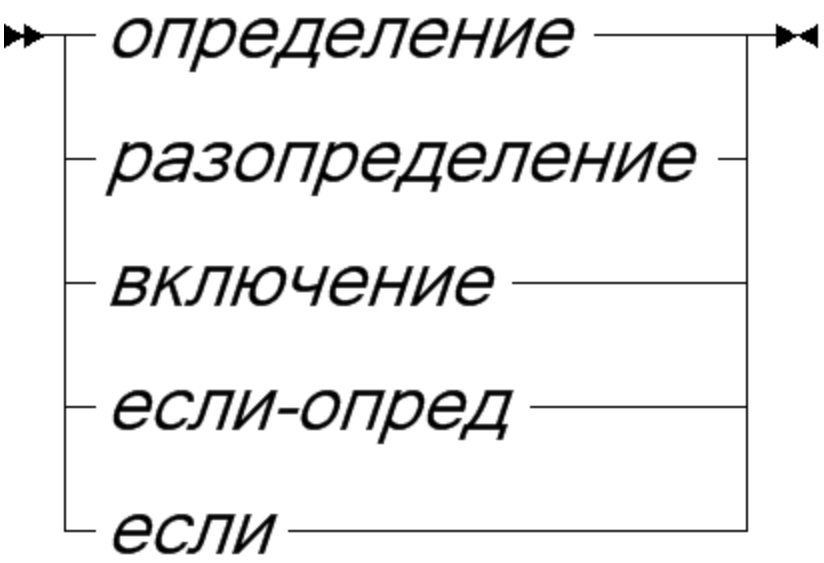
Препроцессор

Синтаксис:

текст

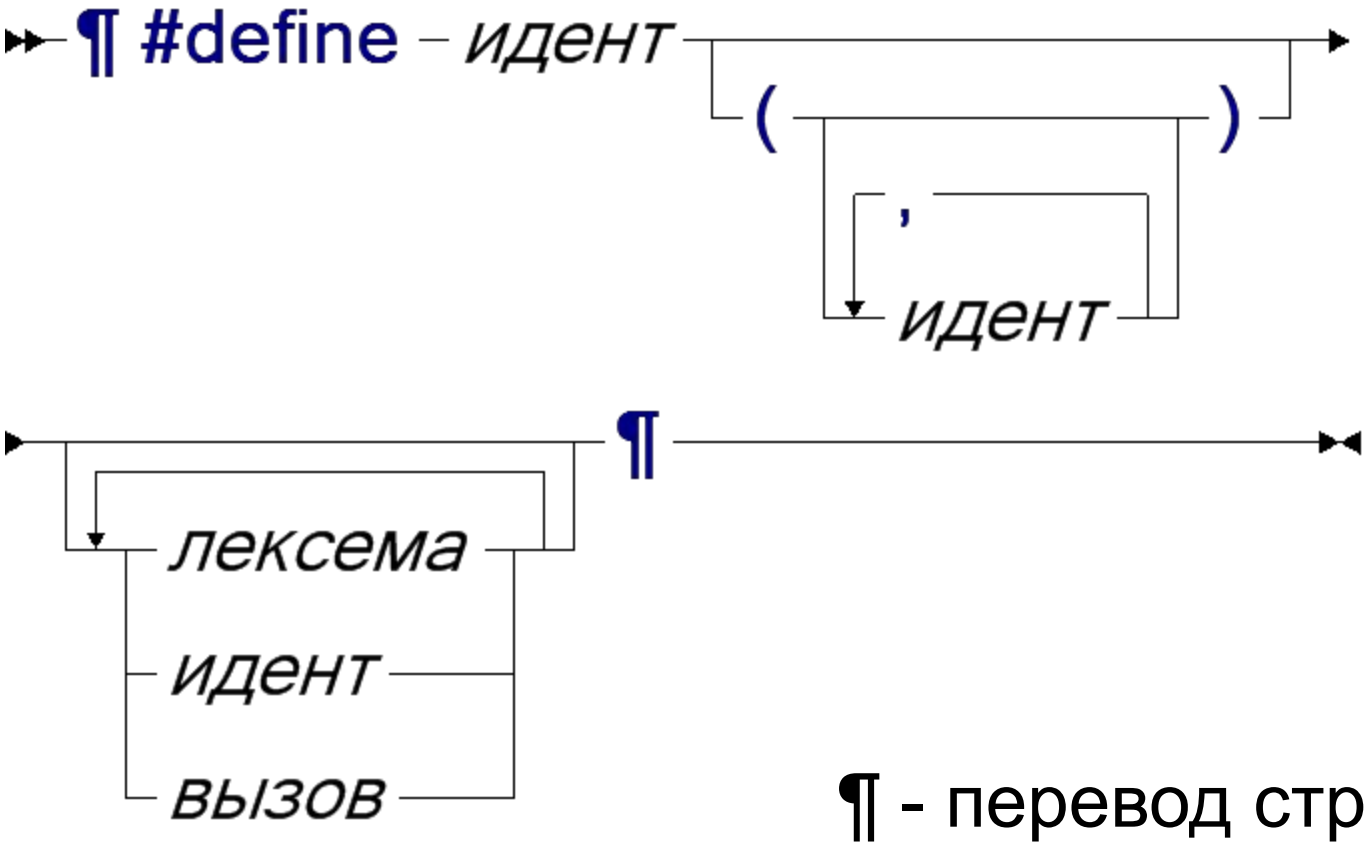


директива



Директивы - определение

Синтаксис *определение*

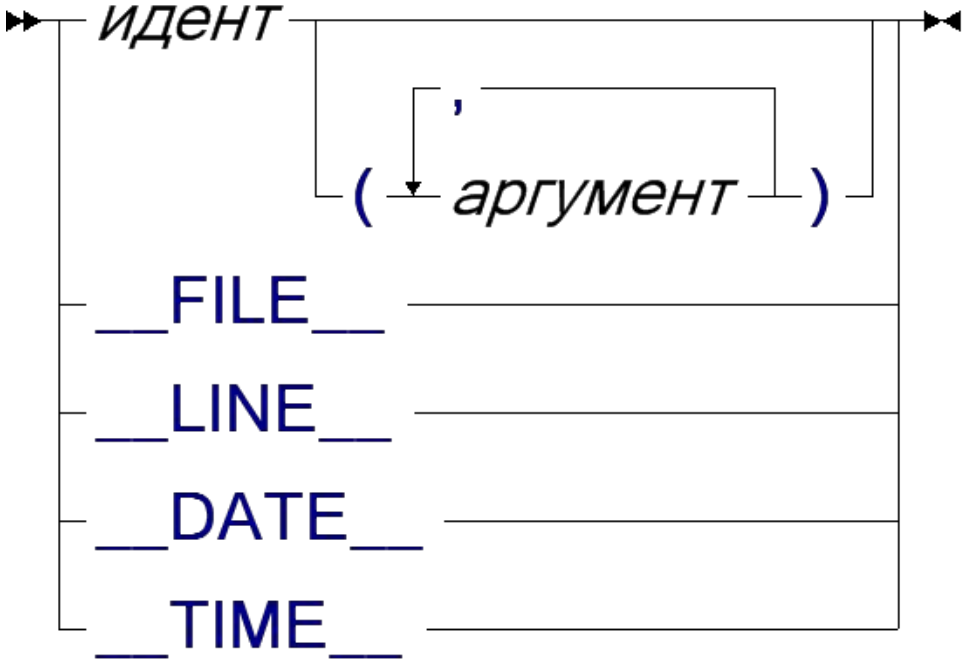


¶ - перевод строки

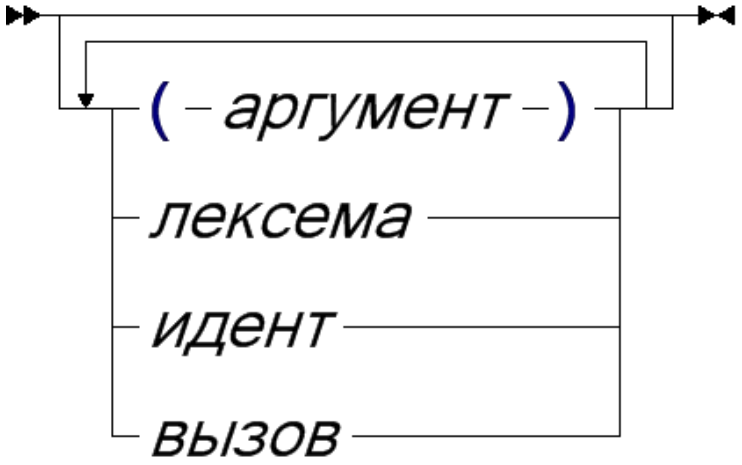
Директивы - вызов

- Синтаксис

вызов



аргумент



Директивы - определение

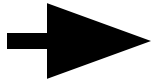
Примеры:

```
#define COOL
#define N 25
#define begin {
#define end }
#define forever for ( ; ; )
#define printnum(n) fprintf(stderr,”%d”,n)
#define printat() fprintf(stderr, \
        ”at: %s [%d]\n”,__FILE__,__LINE__)
```

Директива - вызов

Пример:

```
COOL forever
  begin
     printat();
     printnum(N);
  end
```

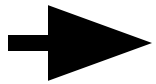


```
for ( ; ; )
   {
      fprintf(stderr,
         ”at: %s [%d]\n”,
         “d:\\temp\\prog.c”,3)
      fprintf(stderr,”%d”,25);
   }
```

Директива - определение

Пример: опасный синтаксис

```
#define max (X, Y) ( X > Y
                  ? X
                  : Y)
max(A,B)
```

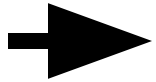


```
                  ? X
                  : Y)
(X, Y) ( X > Y (A,B)
```

Директива - определение

Пример: опасный синтаксис

```
#define max(X, Y) ( X > Y \
                  ? X \
                  : Y)
max(A,B)
```



```
(A > B ? A : B)
```

Директива - определение

Пример: дублирование кода и вычислений

```
#define max(X, Y) ( X > Y ? X : Y)
max( f(A,B) , sqrt(A*A+B*B) )
```

```
(f(A,B) > sqrt(A*A+B*B) ? f(A,B) : sqrt(A*A+B*B))
```

Директива - определение

Пример: рекурсия

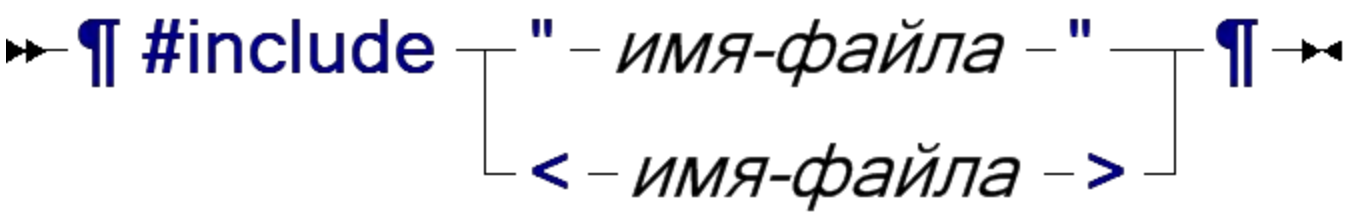
```
#define fact(n) (n==0 ? 1 : (n)*fact(n-1))
fact(10)
```

- зацикливание (10==0 ? 1 : (10)*(10-1==0 ? 1 : (10-1) * (10-1-1==0 ? (10-1-1) * …)))

(Циклов тоже нет. В PL/I – есть.)

Директива - включение

Синтсксис



Директива - включение

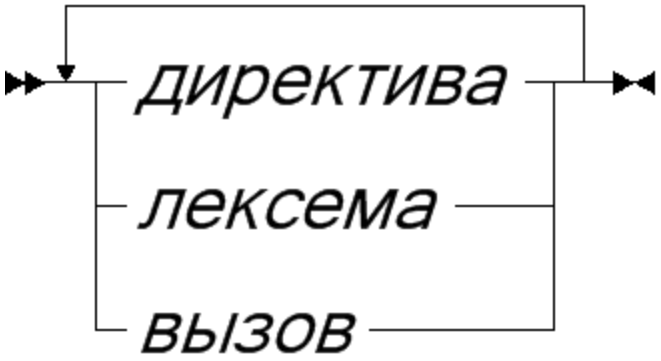
Примеры:

#include “main.h”
#include “..\\include\\person.h”
#include “../include/person.h”
#include “d:\\projects\\dialogs\\form.h”
#include <stdio.h>
#include “stdio.h”

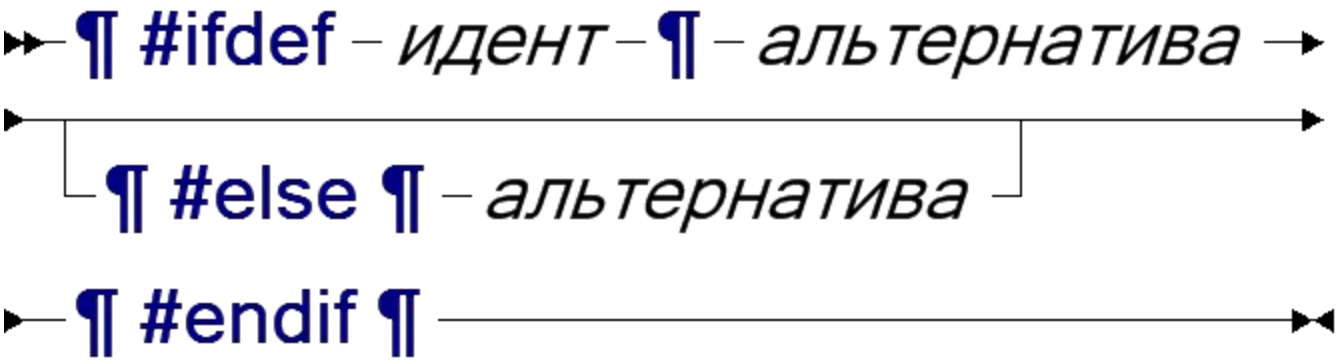
Директива - условное

Синтаксис

Альтернатива



Услов-опред



Директива - условное

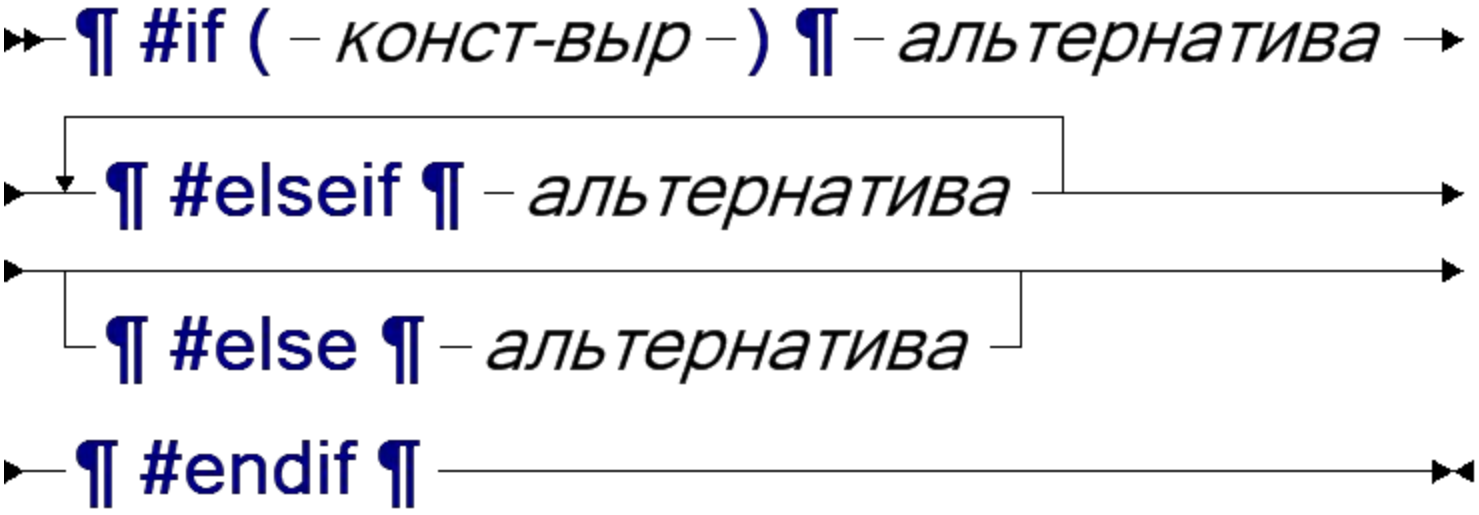
Пример

```
#define SMALL
#ifdef SMALL
#define N 10
#define number short int
#else
#define N 10000
#define number long int
#end if
```

Директива - условное

Синтаксис

Услов-опред



Директива - условное

Пример

```
#define N 18
#define B(k) ((N & ~(k-1)) == 0)
#if (B(8))
#define scale unsigned char
#elseif (B(16))
#define scale unsigned short
#else
#define scale unsigned long
#end if
```

Директивы - условное

Пример: #ifdef – устаревшее.

Эквивалентно:

```
#ifdef A
…
#endif
```

```
#if (defined(A))
…
#endif
```

Допустимы сложные условия с defined

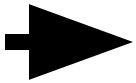
```
#if (defined(A) && !defined(B) || N>3)
…
#endif
```

Операции # и

- конкатенация идентификаторов, # - преобразование в строку

Пример:

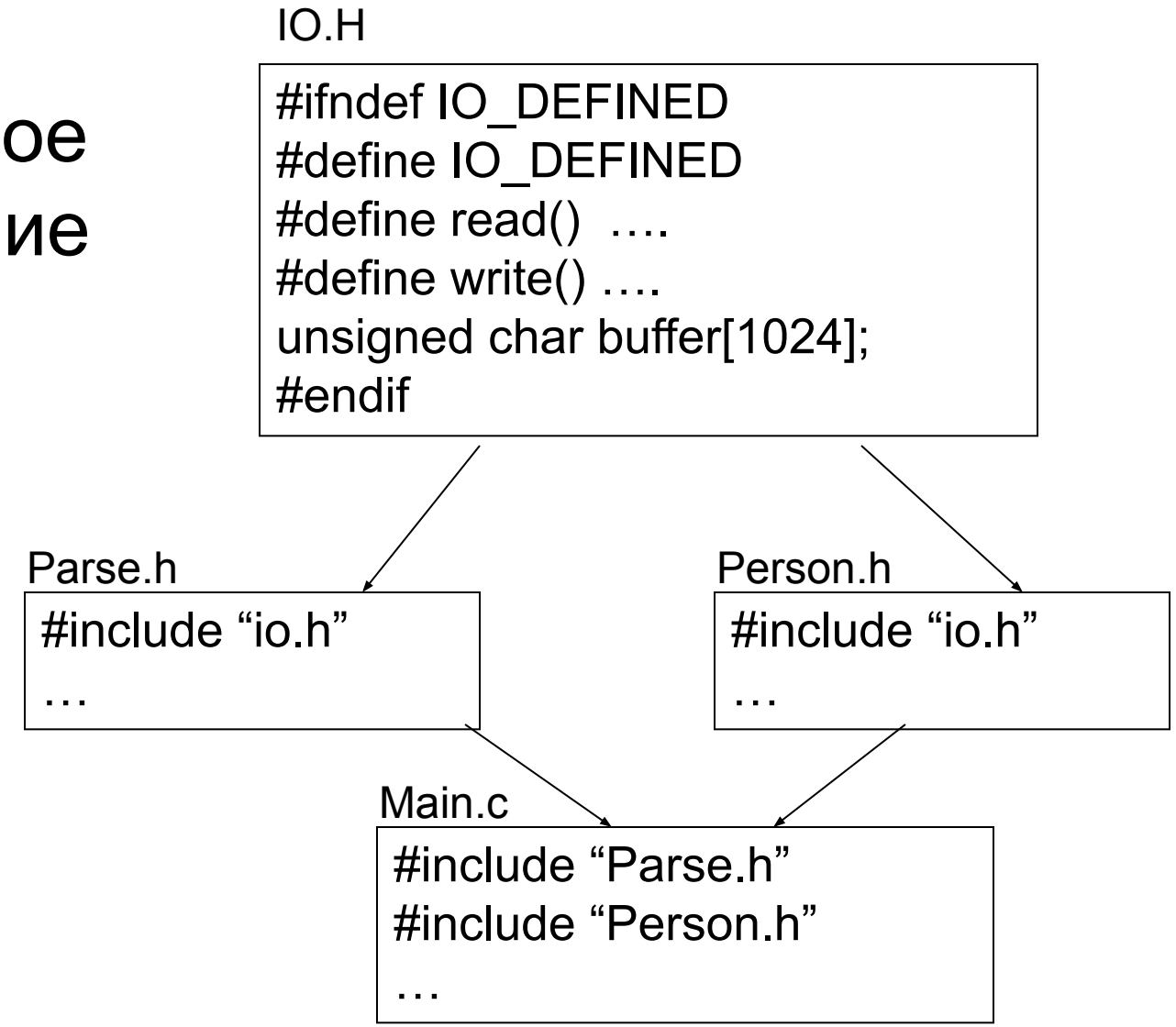
```
#define A(x,y) xy + y
#define B(x,y) x y + y
#define C(x,y) x##y #y
A(a,5)
B(a,5)
C(a,5)
```



```
xy + 5
a 5 + 5
a5 + “5”
```

Типичные использования

- Повторное включение



Типичные использования

- Определение констант

```
#define N 25
#define N2  25*(25-1)
int X[N2-1];
```

Типичные использования

- Условная трансляция

```
switch (code)
{
#ifdef debug
    case codeA :
     …
     break;
#endif
    case codeB :
     …
     break;
    default :
     …
}
```

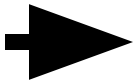
Типичные использования

- Родовые типы

```
#define List(T) \
struct T#List{ \
    T item; \
    struct T#List * next; \
}

List(int) integers;

List(double) reals;
```



```
struct intList{
    int item;
    struct intList * next;
}  integers;

struct doubleList{
    double item;
    struct doubleList * next;
} reals;
```