33. Директивы препроцессора, макросы.
План ответа:
	1. классификация директив препроцессора;
	2. правила, справедливые для всех директив препроцессора;
	3. макросы (простые, с параметрами, с переменным предопределенные);
	4. сравнение макросов с параметрами и функций;
	5. скобки в макросах;
	6. создание длинных макросов;
	7. преопределенные макросы.

1. классификация директив препроцессора;
	Макроопределения
		#define, #undef
	Директива включения файлов
		#include
	Директивы условной компиляции
		#if, #ifdef, #endif и др.
	Остальные директивы (#pragma, #error, #line и др.) используются реже.

2. правила, справедливые для всех директив препроцессора;
	Директивы всегда начинаются с символа "#".
	Любое количество пробельных символов может разделять лексемы в директиве.
	# define N 1000
	Директива заканчивается на символе '\n'.
	Директива заканчивается на символе '\n'.
		#define DISK_CAPACITY (SIDES * \
		 TRACKS_PER_SIDE * \
		 SECTORS_PER_TRACK * \
		 BYTES_PER_SECTOR)

	Директивы могут появляться в любом месте программы.

3. макросы (простые, с параметрами, с переменным предопределенные);
	3.1 простые
		#define идентификатор список-замены

		#define PI 3.14
		#define EOS '\0'
		#define MEM_ERR "Memory allocation error."

		Используются:
		В качестве имен для числовых, символьных и строковых констант.

		Незначительного изменения синтаксиса языка.
		#define BEGIN {
		#define END }
		#define INF_LOOP for(; ;)

		Переименования типов.
		#define BOOL int

		Управления условной компиляцией.

	3.2 с параметрами
		#define идентификатор(x1, x2, ..., xn) список-замены

		Не должно быть пробела между именем макроса и (.
		Список параметров может быть пустым.

		#define MAX(x, y) ((x) > (y) ? (x) : (y))

		Где-то в программе
		i = MAX(j + k, m - n); // i = ((j + k) > (m - n) ? (j +
k) : (m - n));

	3.3 с переменным предопределенные
		#ifndef NDEBUG
		#define DBG_PRINT(s, ...) printf(s, __VA_ARGS__)
		#else

```
#define DBG_PRINT(s, ...) ((void) 0)
#endif
```

4. сравнение макросов с параметрами и функций;

Преимущества
- программа может работать немного быстрее;
- макросы "универсальны".

Недостатки
- скомпилированный код становится больше; n = MAX(i, MAX(j, k)));
- типы аргументов не проверяются;
- нельзя объявить указатель на макрос;
- макрос может вычислять аргументы несколько раз. n = MAX(i++, j);

Общие свойства макросов
- Список-замены макроса может содержать другие макросы.
- Препроцессор заменяет только целые лексемы, не их части.
- Определение макроса остается «известным» до конца файла, в котором этот макрос объявляется.
- Макрос не может быть объявлен дважды, если эти объявление не тождественны.
- Макрос может быть «разопределен» с помощью директивы #undef.

5. скобки в макросах;

Если список-замены содержит операции, он должен быть заключен в скобки.
Если у макроса есть параметры, они должны быть заключены в скобки в список-замены.

```
#define TWO_PI  2 * 3.14

f = 360.0 / TWO_PI;
// f = 360.0 / 2 * 3.14;

#define SCALE(x)  (x * 10)

j = SCALE(i + 1);
// j = (i + 1 * 10);
```

6. создание длинных макросов;

```
// 1
#define ECHO(s) {gets(s); puts(s);}

if (echo_flag)
    ECHO(str);
else
    gets(str);

// 2
#define ECHO(s)  (gets(s), puts(s))
ECHO(str);

// 3
#define ECHO(s) \
do              \
{               \
    gets(s);    \
    puts(s);    \
}               \
while(0)        \
```

7. преопределенные макросы.

Эти идентификаторы нельзя переопределять или отменять директивой undef.
__LINE__ - номер текущей строки (десятичная константа)
__FILE__ - имя компилируемого файла

__DATE__ - дата компиляции
__TIME__ - время компиляции и др.
__func__ - имя функции как строки (GCC only, C99 и не макрос)