34. Директивы препроцессора, условная компиляция, операции # и ##.
План ответа:
	1. классификация директив препроцессора;
	2. правила, справедливые для всех директив препроцессора;
	3. директивы условной компиляции, использование условной компиляции;
	4. директива error;
	5. операция “#”,
	6. операция “##”;
	7. директива pragma (once, pack).

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. директивы условной компиляции, использование условной компиляции;
	Директивы #if, #else, #elif и #endif

	#if defined(OS_WIN)
	...
	#elif defined(OS_LIN)
	...
	#elif defined(OS_MAC)
	...
	#endif

	Возможно, самыми распространенными директивами условной компиляции являют
ся #if, #else, #elif и #endif.
	Они дают возможность в зависимости от значения константного выражения вкл
ючать или исключать те или иные части кода.

	Использование условной компиляции:
		программа, которая должна работать под несколькими операционными
системами;
		программа, которая должна собираться различными компиляторами;
		начальное значение макросов;
		временное выключение кода.

4. директива error;
	#error сообщение

	#if defined(OS_MAC)
	...
	#else
	#error Unsupported OS!
	#endif
5. операция “#”,
	«Операция» # конвертирует аргумент макроса в строковый литерал.

	#define PRINT_INT(n) printf(#n " = %d\n", n)

	#define TEST(condition, ...) ((condition) ? \

```
                printf("Passed test %s\n", #condition) :  \
                                    printf(__VA_ARGS__))
```

Где-то в программе

```
PRINT_INT(i / j);
// printf("i/j" " = %d", i/j);

TEST(voltage <= max_voltage,
      "Voltage %d exceed %d", voltage, max_voltage);
```

6. операция “##”;

«Операция» ## объединяет две лексемы в одну.

```
#define MK_ID(n)        i##n
```

Где-то в программе

```
int MK_ID(1), MK_ID(2);
// int  i1, i2;
```

Более содержательный пример

```
#define GENERAL_MAX(type)          \
type type##_max(type x, type y)    \
{                                  \
    return x > y ? x : y;          \
}
```

7. директива pragma (once, pack).

Директива #pragma позволяет добиться от компилятора специфичного поведения.

Каждая реализация C поддерживает некоторые функции, уникальные для хост-компьютера или операционной системы.

Некоторые программы, например, должны осуществлять точный контроль над областями памяти, где размещаются данные, или контролировать способ получения параметров определенными функциями.

Директивы #pragma предлагают каждому компилятору возможность предоставлять функции для конкретного компьютера и операционной системы, сохраняя общую совместимость.