1.What does static variable mean?

Ans: Static variables are the variables which retain their values between the function calls. **They are initialized only once their scope is within the function** in which they are defined.

2. What is a pointer?

Ans: Pointers are variables which **stores the address of another variable**. That variable may be a scalar (including another pointer), or an aggregate (array or structure). The pointed-to object may be part of a larger object, such as a field of a structure or an element in an array.

3. What are the uses of a pointer?

Ans: Pointer is used in the following cases

i) It is used to **access array elements**

ii) It is used for **dynamic memory allocation**.

iii) It is used in **Call by reference**

iv) It is used in data structures like trees, graph, linked list etc.

4. What is a structure?

Ans: Structure constitutes a super data type which represents several different data types in a single unit. A structure **can be initialized if it is static or global**.

5. What is a union?

Ans: Union is a collection of **heterogeneous data type** but it uses efficient memory utilization technique by allocating enough memory to hold the largest member. Here a single area of memory contains values of different types at different time. A union can never be initialized.

6. What are the differences between structures and union?

Ans: A **structure** variable contains each of the named members, and its size is large enough to hold all the members. **Structure elements are of same size**.
        A **union** contains one of the named members at a given time and is large enough to hold the largest member. **Union element can be of different sizes**.

7. What are the differences between structures and arrays?

Ans: Structure is a collection of heterogeneous data type but array is a collection of homogeneous data types.

Array

1. It is a collection of data items of same data type.
2. It has declaration only
3. There is no keyword.
4. Array name represent the address of the starting element.

Structure

1-It is collection of data items of different data type.
2- It has declaration and definition
3- keyword struct is used
4-Structure name is known as tag it is the short hand notation of the declaration.

8. In header files whether functions are declared or defined?

Ans: Functions are declared within header file. That is function prototypes exist in a header file, not function bodies. They are defined in library (lib).

9. What are the differences between Malloc () and Calloc ()?
Ans: **Malloc ()**
1-Malloc takes one argument Malloc(a);where a number of bytes
2-memory allocated contains garbage values
Calloc()
1-Calloc takes two arguments Calloc(b,c) where b no of object and c size of object
2-It initializes the contains of block of memory to zerosMalloc takes one argument, memory allocated contains garbage values.
         It allocates contiguous memory locations. Calloc takes two arguments, memory allocated contains all zeros, and the memory allocated is not contiguous.

10. What are macros? What are its advantages and disadvantages?
Ans: Macros are abbreviations for lengthy and frequently used statements. When a macro is called the entire code is substituted by a single line though the macro definition is of several lines.
      The **advantage** of macro is that it reduces the time taken for control transfer as in case of
function.
       The **disadvantage** of it is here the entire code is substituted so the program becomes lengthy if a macro is called several times.

11. Difference between pass by reference and pass by value?
Ans: Pass by reference passes a **pointer to the value**. This allows the callee to modify the variable directly.
         Pass by value gives **a copy of the value to the callee**. This allows the callee to modify the value without modifying the variable. (In other words, the callee simply cannot modify the variable, since it lacks a reference to it.)

12. What is static identifier?
Ans: A file-scope variable that is declared static is visible only to functions within that file. A function-scope or block-scope variable that is declared as static is visible only within that scope. Furthermore, static variables only have a single instance. In the case of function- or block-scope variables, this means that the variable is not "automatic" and thus retains its value across function invocations.

13. Where is the auto variables stored?
Ans: Auto variables can be **stored anywhere**, so long as recursion works. Practically, they're stored on the stack. It is not necessary that always a stack exist. You could theoretically allocate function invocation records from the heap.

14. Where does global, static, and local, register variables, free memory and C Program instructions get stored?
Ans: Global: Wherever the linker puts them. Typically the "BSS segment" on many platforms.
Static: Again, wherever the linker puts them. Often, they're intermixed with the globals. The only difference between globals and statics is whether the linker will resolve the symbols across compilation units.
Local: Typically on the stack, unless the variable gets register allocated and never spills.

Register: Nowadays, these are equivalent to "Local" variables. They live on the stack unless they get register-allocated.

15. Difference between arrays and linked list?

Ans: An array is **a repeated pattern of variables in contiguous storage**. A linked list is a **set of structures scattered through memory**, held together by pointers in each element that point to the next element. With an array, we can (on most architectures) move from one element to the next by adding a fixed constant to the integer value of the pointer. With a linked list, there is a "next" pointer in each structure which says what element comes next.

16. What are enumerations?

Ans: They are a list of named integer-valued constants.

   Example: enum color { black , orange=4,yellow, green, blue, violet };

This declaration defines the symbols "black", "orange", "yellow", etc. to have the values "1," "4," "5," ... etc.

     The difference between an enumeration and a macro is that the enum actually declares a type, and therefore can be type checked.

17. Describe about storage allocation and scope of global, extern, static, local and register variables?

Ans: Globals have application-scope. They're available in any compilation unit that includes an appropriate declaration (usually brought from a header file). They're stored wherever the linker puts them, usually a place called the "BSS segment."

Extern: This is essentially "global."

Static: Stored the same place as globals, typically, but only available to the compilation unit that contains them. If they are block-scope global, only available within that block and its subblocks.

Local: Stored on the stack, typically. Only available in that block and its subblocks. (Although pointers to locals can be passed to functions invoked from within a scope where that local is valid.)

Register: See tirade above on "local" vs. "register." The only difference is that the C compiler will not let you take the address of something you've declared as "register."

18. What are register variables? What are the advantages of using register variables?

Ans: If a variable is declared with a register storage class, it is known as register variable. The register variable is stored in the cpu register instead of main memory. Frequently used variables are declared as register variable as it's access time is faster.

19. What is the use of typedef?

Ans: The typedef help in easier modification when the programs are ported to another machine. A descriptive new name given to the existing data type may be easier to understand the code.

20. Can we specify variable field width in a scanf () format string? If possible how?

Ans: All field widths are variable with scanf (). You can specify a maximum field width for a

given field by placing an integer value between the '%' and the field type specifier. (e.g. %64s). Such a specifier will still accept a narrower field width. The one exception is %#c (where # is an integer). This reads EXACTLY # characters, and it is the only way to specify a fixed field width with scanf ()

21. Out of fgets() and gets() which function is safe to use and why?

Ans: fgets () is safer than gets (), because we can specify a maximum input length. Neither one is completely safe, because the compiler can't prove that programmer won't overflow the buffer he pass to fgets ().

22. Difference between strdup and strcpy?

Ans: Both copy a string. strcpy wants a buffer to copy into. strdup allocates a buffer using Malloc ().
Unlike strcpy (), strdup () is not specified by ANSI.

23. What is recursion?

Ans: A recursion function is one **which calls itself either directly or indirectly** it must halt at a definite point to avoid infinite recursion.

24. Differentiate between for loop and a while loop? What are it uses?

Ans: For executing a set of statements fixed number of times we use for loop while when the number of iterations to be performed is not known in advance we use while loop.

25. What is storage class? What are the different storage classes in C?

Ans: Storage class is an attribute that changes the behaviour of a variable. It controls the lifetime, scope and linkage. The storage classes in c are auto, register, and extern, static, typedef.

26. What the advantages of using Unions?

Ans: When the C compiler is allocating memory for unions it will always reserve enough room for the largest member.

27. What is the difference between Strings and Arrays?

Ans: String is a sequence of characters ending with NULL .it can be treated as a one dimensional array of characters terminated by a NULL character.

28. What is a far pointer? Where we use it?

Ans: In large data model (compact, large, huge) the address B0008000 is acceptable because in these model all pointers to data are 32bits long. If we use small data model (tiny, small, medium) the above address won't work since in these model each pointer is 16bits long. If we are working in a small data model and want to access the address B0008000 then we use far pointer. Far pointer is always treated as a 32bit pointer and contains a segment address and offset address both of 16bits each. Thus the address is represented using segment: offset format B000h:8000h. For any given memory address there are many possible far address segment: offset pair. The segment register contains the address where the segment begins and offset register contains the offset of data/code from where segment begins.

29. What is a huge pointer?

Ans: Huge pointer is 32bit long containing segment address and offset address. Huge pointers are normalized pointers so for any given memory address there is only one possible huge address segment: offset pair. Huge pointer arithmetic is doe with calls to special subroutines so its arithmetic slower than any other pointers.

30. What is a normalized pointer, how do we normalize a pointer?

Ans: It is a 32bit pointer, which has as much of its value in the segment register as possible. Since a segment can start every 16bytes so the offset will have a value from 0 to F. for normalization convert the address into 20bit address then use the 16bit for segment address and 4bit for the offset address. Given a pointer 500D: 9407,we convert it to a 20bitabsolute address 549D7,Which then normalized to 549D:0007.

31. What is near pointer?

Ans: A near pointer is 16 bits long. It uses the current content of the CS (code segment) register (if the pointer is pointing to code) or current contents of DS (data segment) register (if the pointer is pointing to data) for the segment part, the offset part is stored in a 16 bit near pointer. Using near pointer limits the data/code to 64kb segment.

32. In C, why is the void pointer useful? When would you use it?

Ans: The void pointer is useful because it is a generic pointer that any pointer can be cast into and back again without loss of information.

33. What is a NULL Pointer? Whether it is same as an uninitialized pointer?

Ans: Null pointer is a pointer which points to nothing but uninitialized pointer may point to anywhere.

34. Are pointers integer?

Ans: No, pointers are not integers. A pointer is an address. It is a positive number.

35. What does the error 'Null Pointer Assignment' means and what causes this error?

Ans: As null pointer points to nothing so accessing a uninitialized pointer or invalid location may cause an error.

36. What is generic pointer in C?

Ans: In C void* acts as a generic pointer. When other pointer types are assigned to generic pointer, conversions are applied automatically (implicit conversion).

37. Are the expressions arr and &arr same for an array of integers?

Ans: Yes for array of integers they are same.

38. IMP>How pointer variables are initialized?

Ans: Pointer variables are initialized by one of the following ways.

I. Static memory allocation
II. Dynamic memory allocation

39. What is static memory allocation?
Ans: Compiler allocates memory space for a declared variable. By using the address of operator, the reserved address is obtained and this address is assigned to a pointer variable. This way of assigning pointer value to a pointer variable at compilation time is known as static memory allocation.

40. What is dynamic memory allocation?
Ans: A dynamic memory allocation uses functions such as Malloc () or Calloc () to get memory dynamically. If these functions are used to get memory dynamically and the values returned by these function are assigned to pointer variables, such a way of allocating memory at run time is known as dynamic memory allocation.

41. What is the purpose of realloc?
Ans: It increases or decreases the size of dynamically allocated array. The function realloc (ptr, n) uses two arguments. The first argument ptr is a pointer to a block of memory for which the size is to be altered. The second argument specifies the new size. The size may be increased or decreased. If sufficient space is not available to the old region the function may create a new region.

42. What is pointer to a pointer?
Ans: If a pointer variable points another pointer value. Such a situation is known as a pointer to a pointer.
Example:
int *p1,**p2,v=10;
P1=&v; p2=&p1;
Here p2 is a pointer to a pointer.

43. What is an array of pointers?
Ans: if the elements of an array are addresses, such an array is called an array of pointers.

44. Difference between linker and linkage?
Ans: Linker converts an object code into an executable code by linking together the necessary built in functions. The form and place of declaration where the variable is declared in a program determine the linkage of variable.

45. Is it possible to have negative index in an array?
Ans: Yes it is possible to index with negative value provided there are data stored in this location. Even if it is illegal to refer to the elements that are out of array bounds, the compiler will not produce error because C has no check on the bounds of an array.

46. Why is it necessary to give the size of an array in an array declaration?
Ans: When an array is declared, the compiler allocates a base address and reserves enough

space in memory for all the elements of the array. The size is required to allocate the required space and hence size must be mentioned.

47. What modular programming?
Ans: If a program is large, it is subdivided into a number of smaller programs that are called modules or subprograms. If a complex problem is solved using more modules, this approach is known as modular programming.

48. What is a function?
Ans: A large program is subdivided into a number of smaller programs or subprograms. Each subprogram specifies one or more actions to be performed for the larger program. Such sub programs are called functions.

49. What is an argument?
Ans: An argument is an entity used to pass data from the calling to a called function.

50. What are built in functions?
Ans: The functions that are predefined and supplied along with the compiler are known as built-in functions. They are also known as library functions.

51. Difference between formal argument and actual argument?
Ans: Formal arguments are the arguments available in the function definition. They are preceded by their own data type. Actual arguments are available in the function call. These arguments are given as constants or variables or expressions to pass the values to the function.

52. Is it possible to have more than one main() function in a C program ?
Ans: The function main () can appear only once. The program execution starts from main.

53. What is the difference between an enumeration and a set of pre-processor # defines?
Ans: There is hardly any difference between the two, except that #defines has a global effect (throughout the file) whereas an enumeration can have an effect local to the block if desired. Some advantages of enumeration are that the numeric values are automatically assigned whereas in #define we have to explicitly define them. A disadvantage is that we have no control over the size of enumeration variables.

54. How are Structure passing and returning implemented by the complier?
Ans: When structures are passed as argument to functions, the entire structure is typically pushed on the stack. To avoid this overhead many programmer often prefer to pass pointers to structure instead of actual structures. Structures are often returned from functions in a location pointed to by an extra, compiler-supported 'hidden' argument to the function.

55. IMP>what is the similarity between a Structure, Union and enumeration?
Ans: All of them let the programmer to define new data type.

56. Can a Structure contain a Pointer to itself?
Ans: Yes such structures are called self-referential structures.

57. How can we read/write Structures from/to data files?

Ans: To write out a structure we can use fwrite () as Fwrite (&e, sizeof (e), 1, fp); Where e is a structure variable. A corresponding fread () invocation can read the structure back from file. calling fwrite () it writes out sizeof (e) bytes from the address &e. Data files written as memory images with fwrite (), however, will not be portable, particularly if they contain floating point fields or Pointers. This is because memory layout of structures is machine and compiler dependent. Therefore, structures written as memory images cannot necessarily be read back by programs running on other machine, and this is the important concern if the data files you're writing will ever be interchanged between machines.

58. Write a program which employs Recursion?

Ans: int fact(int n) { return n > 1 ? n * fact(n – 1) : 1; }

59. Write a program which uses Command Line Arguments?

Ans: #include
void main(int argc,char *argv[])
{
int i;
clrscr();
for(i=0;i
printf("\n%d",argv[i]);
}

60. Difference between array and pointer?

Ans: Array

1- Array allocates space automatically

2- It cannot be resized

3- It cannot be reassigned

4- sizeof (arrayname) gives the number of bytes occupied by the array.

Pointer

1-Explicitly assigned to point to an allocated space.

2-It can be sized using realloc()

3-pointer can be reassigned.

4-sizeof (p) returns the number of bytes used to store the pointer variable p.

61. What do the 'c' and 'v' in argc and argv stand for?

Ans: The c in argc(argument count) stands for the number of command line argument the program is invoked with and v in argv(argument vector) is a pointer to an array of character string that contain the arguments.

62. IMP>what are C tokens?

Ans: There are six classes of tokens: identifier, keywords, constants, string literals, operators and other separators.

63. What are C identifiers?

Ans: These are names given to various programming element such as variables, function,

arrays. It is a combination of letter, digit and underscore. It should begin with letter. Backspace is not allowed.

64. Difference between syntax vs logical error?

Ans: Syntax Error

1-These involves validation of syntax of language.

2-compiler prints diagnostic message.

Logical Error

1-logical error are caused by an incorrect algorithm or by a statement mistyped in such a way that it doesn't violet syntax of language.

2-difficult to find.

65. What is preincrement and post increment?

Ans: ++n (pre increment) increments n before its value is used in an assignment operation or any expression containing it. n++ (post increment) does increment after the value of n is used.

66. Write a program to interchange 2 variables without using the third one.

Ans:

a ^= b; ie a=a^b

b ^= a; ie b=b^a;

a ^= b ie a=a^b;

here the numbers are converted into binary and then xor operation is performed.

You know, you're just asking "have you seen this overly clever trick that's not worth applying on modern architectures and only really applies to integer variables?"

67. What is the maximum combined length of command line arguments including the space between adjacent arguments?

Ans: It depends on the operating system.

68. What are bit fields? What is the use of bit fields in a Structure declaration?

Ans: A bit field is a set of adjacent bits within a single implementation based storage unit that we will call a "word".

The syntax of field definition and access is based on structure.

Struct {

unsigned int k :1;

unsigned int l :1;

unsigned int m :1;

}flags;

the number following the colon represents the field width in bits.Flag is a variable that contains three bit fields.

69. What is a preprocessor, what are the advantages of pre-processor?

Ans: A pre-processor processes the source code program before it passes through the compiler.

1- a pre-processor involves the readability of program

2- It facilitates easier modification

3- It helps in writing portable programs
4- It enables easier debugging
5- It enables testing a part of program
6- It helps in developing generalized program

70. What are the facilities provided by pre-processor?
Ans:
1-file inclusion
2-substitution facility
3-conditional compilation

71. What are the two forms of #include directive?
Ans:
1.#include"filename"
2.#include
the first form is used to search the directory that contains the source file. If the search fails in the home directory it searches the implementation defined locations. In the second form ,the pre-processor searches the file only in the implementation defined locations.

72. How would you use the functions randomize () and random ()?
Ans:
Randomize() initiates random number generation with a random value.
Random () generates random number between 0 and n-1;

73. What do the functions atoi (), itoa () and gcvt() do?
Ans:
atoi() is a macro that converts integer to character.
itoa () It converts an integer to string
gcvt () It converts a floating point number to string

74. How would you use the functions fseek(), freed(), fwrite() and ftell()?
Ans:
fseek(f,1,i) Move the pointer for file f a distance 1 byte from location i.
fread(s,i1,i2,f) Enter i2 dataitems,each of size i1 bytes,from file f to string s.
fwrite(s,i1,i2,f) send i2 data items,each of size i1 bytes from string s to file f.
ftell(f) Return the current pointer position within file f. The data type returned for functions fread,fseek and fwrite is int and ftell is long int.

75. What is the difference between the functions memmove() and memcpy()?
Ans: The arguments of memmove() can overlap in memory. The arguments of memcpy() cannot.

76. What is a file?
Ans: A file is a region of storage in hard disks or in auxiliary storage devices.It contains bytes of information .It is not a data type.

77. IMP>what are the types of file?
Ans: Files are of two types
1-high level files (stream oriented files) :These files are accessed using library functions
2-low level files(system oriented files) :These files are accessed using system calls

78. IMP>what is a stream?
Ans: A stream is a source of data or destination of data that may be associated with a disk or other I/O device. The source stream provides data to a program and it is known as input stream. The destination stream eceives the output from the program and is known as output stream.

79. What is meant by file opening?
Ans: The action of connecting a program to a file is called opening of a file. This requires creating an I/O stream before reading or writing the data.

80. What is FILE?
Ans: FILE is a predefined data type. It is defined in stdio.h file.

81. What is a file pointer?
Ans: The pointer to a FILE data type is called as a stream pointer or a file pointer. A file pointer points to the block of information of the stream that had just been opened.

82. How is fopen()used ?
Ans: The function fopen() returns a file pointer. Hence a file pointer is declared and it is assigned as
FILE *fp;
fp= fopen(filename,mode);
filename is a string representing the name of the file and the mode represents:
"r" for read operation
"w" for write operation
"a" for append operation
"r+","w+","a+" for update operation

83How is a file closed ?
Ans: A file is closed using fclose() function
Eg. fclose(fp);
Where fp is a file pointer.

84. What is a random access file?
Ans:
A file can be accessed at random using fseek() function
fseek(fp,position,origin);
fp file pointer
position number of bytes offset from origin
origin 0,1 or 2 denote the beginning ,current position or end of file respectively.

85. What is the purpose of ftell ?

Ans: The function ftell() is used to get the current file represented by the file pointer.

ftell(fp);

returns a long integer value representing the current file position of the file pointed by the file pointer fp.If an error occurs ,-1 is returned.

86. What is the purpose of rewind() ?

Ans: The function rewind is used to bring the file pointer to the beginning of the file.

Rewind(fp);

Where fp is a file pointer.Also we can get the same effect by

feek(fp,0,0);

87. Difference between a array name and a pointer variable?

Ans: A pointer variable is a variable where as an array name is a fixed address and is not a variable. A

pointer variable must be initialized but an array name cannot be initialized. An array name being a constant value , ++ and — operators cannot be applied to it.

88. Represent a two-dimensional array using pointer?

Ans:

Address of a[I][j] Value of a[I][j]

&a[I][j]

or

a[I] + j

or

*(a+I) + j

*&a[I][j] or a[I][j]

or

*(a[I] + j)

or

*(* (a+I) +j)

89. Difference between an array of pointers and a pointer to an array?

Ans:

Array of pointers

1- Declaration is: data_type *array_name[size];

2-Size represents the row size.

3- The space for columns may be dynamically

Pointers to an array

1-Declaration is data_type (*array_name)[size];

2-Size represents the column size.

90. Can we use any name in place of argv and argc as command line arguments ?
Ans: yes we can use any user defined name in place of argc and argv;

91. What are the pointer declarations used in C?
Ans:
1- Array of pointers, e.g , int *a[10]; Array of pointers to integer
2-Pointers to an array,e.g , int (*a)[10]; Pointer to an array of into
3-Function returning a pointer,e.g, float *f( ) ; Function returning a pointer to float
4-Pointer to a pointer ,e.g, int **x; Pointer to apointer to int
5-pointer to a data type ,e.g, char *p; pointer to char

92. Differentiate between a constant pointer and pointer to a constant?
Ans:
const char *p; //pointer to a const character.
char const *p; //pointer to a const character.
char * const p; //const pointer to a char variable.
const char * const p; // const pointer to a const character.

93. Is the allocated space within a function automatically deallocated when the function returns?
Ans: No pointer is different from what it points to .Local variables including local pointers variables in a function are deallocated automatically when function returns.,But in case of a local pointer variable ,deallocation means that the pointer is deallocated and not the block of memory allocated to it. Memory dynamically allocated always persists until the allocation is freed
or the program terminates.

94. Discuss on pointer arithmetic?
Ans:
1- Assignment of pointers to the same type of pointers.
2- Adding or subtracting a pointer and an integer.
3-subtracting or comparing two pointer.
4-incrementing or decrementing the pointers pointing to the elements of an array. When a pointer
to an integer is incremented by one , the address is incremented by two. It is done automatically
by the compiler.
5-Assigning the value 0 to the pointer variable and comparing 0 with the pointer. The pointer having address 0 points to nowhere at all.

95. What is the invalid pointer arithmetic?
Ans:
i) adding ,multiplying and dividing two pointers.
ii) Shifting or masking pointer.
iii) Addition of float or double to pointer.
iv) Assignment of a pointer of one type to a pointer of another type ?

96. What are the advantages of using array of pointers to string instead of an array of strings?
Ans:
i) Efficient use of memory.
ii) Easier to exchange the strings by moving their pointers while sorting.

97. Are the expressions *ptr ++ and ++ *ptr same?
Ans: No,*ptr ++ increments pointer and not the value pointed by it. Whereas ++ *ptr increments the value being pointed to by ptr.

98. What would be the equivalent pointer expression foe referring the same element as a[p][q][r][s] ?
Ans : *(* (* (* (a+p) + q) + r) + s)

99. Are the variables argc and argv are always local to main?
Ans: Yes they are local to main.

100. Can main () be called recursively?
Ans: Yes any function including main () can be called recursively.

101. IMP>Can we initialize unions?
Ans: ANSI Standard C allows an initializer for the first member of a union. There is no standard way
of initializing any other member (nor, under a pre-ANSI compiler, is there generally any way of
initializing a union at all).

102. What's the difference between these two declarations?
Ans: struct x1 { … };
typedef struct { … } x2;
The first form declares a structure tag; the second declares a typedef. The main difference is that the second declaration is of a slightly more abstract type.its users don't necessarily know that it is a structure, and the keyword struct is not used when declaring instances of it.

103. Why doesn't this code: a[i] = i++; work?
Ans: The subexpression i++ causes a side effect.it modifies i's value.which leads to undefined
behavior since i is also referenced elsewhere in the same expression.

104.WHy doesn't struct x { … };
x thestruct;
work?
Ans:
C is not C++. Typedef names are not automatically generated for structure tags.

105. Why can't we compare structures?
Ans:
There is no single, good way for a compiler to implement structure comparison which is consistent with C's low-level flavor. A simple byte-by-byte comparison could founder on random bits present in unused "holes" in the structure (such padding is used to keep the alignment of later fields correct). A field-by-field comparison might require unacceptable amounts of repetitive code for large structures.

106. How are structure passing and returning implemented?
Ans: When structures are passed as arguments to functions, the entire structure is typically pushed on
the stack, using as many words as are required. Some compilers merely pass a pointer to the structure, though they may have to make a local copy to preserve pass-by-value semantics. Structures are often returned from functions in a location pointed to by an extra,compiler-supplied "hidden" argument to the function. Some older compilers used a special,static location for structure returns, although this made structure-valued functions non-reentrant, which ANSI C disallows.