Computer memory is represented as a sequence of bytes, grouped into words. Bytes have unique addresses or indices. The size of a word and byte determines the amount of memory that can be accessed. In C, a pointer is a variable that stores the memory address of another variable, which can also be a pointer.

In C, pointers allow retrieving the value they point to using the unary * operator. For example, int *p; and int x = *p; retrieves the value pointed to by p. The memory address of a variable can be obtained using the unary ampersand (&) operator, such as int *p = &x; to store the address of x in p.

Pointer arithmetic allows adjusting the location a pointer points to. For instance, if a pointer pc points to the first element of an array, executing pc += 3; will make pc point to the fourth element. Moreover, pointers can be dereferenced using array notation

In C, all arguments to a function are passed by value, meaning that a copy of the argument is made and used within the function. Modifying the local copy of a variable inside a function does not affect the original variable outside of the function.However, when an array is passed as an argument to a function, what is actually passed is a pointer to the first element of the array.

C allows the creation of arrays of pointers, such as `int *a[5]`. Arrays of pointers are especially useful when working with strings. For instance, in C's support for command line arguments, the `main` function can be defined as `int main(int argc, char *argv[])`, where `argv` is an array of character pointers representing the command line arguments, and `argc` indicates the length of the array.C also supports the declaration of multi-dimensional arrays.

In C, to define an instance of a structure called `circle`, we write `struct circle c;`. Structures can also be initialized with values using the syntax `struct circle c = {12, 23, 5};`. An automatic or local structure variable can be initialized by a function call. For example, `struct circle c = circle_init();` initializes the structure variable `c` with values returned by the `circle_init()` function.Multiple instances of a structure can be declared and defined in a single statement.

In C, programmers can use pointers to functions, enabling functions to be passed as arguments to other functions. This allows for increased flexibility and parameterization of algorithms. For instance, a sorting algorithm can be designed to accept a pointer to a comparison function.

In C, a structure member can be accessed using the dot notation, such as `structname.member`, for example: `pt.x`. However, comparison operations between structures, like `pt1 > pt2`, are not defined in C.Pointers to structures can be defined using the syntax `struct circle *pc`. When using a pointer to a structure, member access can be achieved with the dot operator, but it can look cumbersome, as in `(*pc).x`.