Main function

Every C program coded to run in a hosted execution environment contains the definition (not the prototype) of a function named **main**, which is the designated start of the program.

`int main (void) { body }`	(1)
`int main (int argc, char *argv[]) { body }`	(2)
`/* another implementation-defined signature */` (since C99)	(3)

Parameters

argc -	Non-negative value representing the number of arguments passed to the program from the environment in which the program is run.
argv -	Pointer to the first element of an array of `argc + 1` pointers, of which the last one is null and the previous ones, if any, point to strings that represent the arguments passed to the program from the host environment. If `argv[0]` is not a null pointer (or, equivalently, if `argc` > 0), it points to a string that represents the program name, which is empty if the program name is not available from the host environment.

The names `argc` and `argv` stand for "argument count" and "argument vector", and are traditionally used, but other names may be chosen for the parameters, as well as different but equivalent declarations of their type: `int main(int ac, char** av)` is equally valid.

A common implementation-defined form of main is `int main(int argc, char *argv[], char *envp[])`, where a third argument, of type `char**`, pointing at an array of pointers to the *execution environment variables* (https://pubs.opengroup.org/onlinepubs/9699919799/functions/exec.html) , is added.

Return value

If the return statement is used, the return value is used as the argument to the implicit call to `exit()` (see below for details). The values zero and EXIT_SUCCESS indicate successful termination, the value EXIT_FAILURE indicates unsuccessful termination.

Explanation

The `main` function is called at program startup, after all objects with static storage duration are initialized. It is the designated entry point to a program that is executed in a *hosted* environment (that is, with an operating system). The name and type of the entry point to any *freestanding* program (boot loaders, OS kernels, etc) are implementation-defined.

The parameters of the two-parameter form of the main function allow arbitrary multibyte character strings to be passed from the execution environment (these are typically known as *command line arguments*). The pointers `argv[1] .. argv[argc-1]` point at the first characters in each of these strings. `argv[0]` (if non-null) is the pointer to the initial character of a null-terminated multibyte string that represents the name used to invoke the program itself (or, if this is not supported by the host environment, `argv[0][0]` is guaranteed to be zero).

If the host environment cannot supply both lowercase and uppercase letters, the command line arguments are converted to lowercase.

The strings are modifiable, and any modifications made persist until program termination, although these modifications do not propagate back to the host environment: they can be used, for example, with `strtok`.

The size of the array pointed to by `argv` is at least `argc+1`, and the last element, `argv[argc]`, is guaranteed to be a null pointer.

The `main` function has several special properties:

1) A prototype for this function cannot be supplied by the program.
2) If the return type of the main function is compatible with `int`, then the return from the initial call to main (but not the return from any subsequent, recursive, call) is equivalent to executing the `exit` function, with the value that the main function is returning passed as the argument (which then calls the functions registered with `atexit`, flushes and closes all streams, and deletes the files created with `tmpfile`, and returns control to the execution environment).
3)

If the main function executes a `return` that specifies no value or, which is the same, reaches the terminating `}` without executing a `return`, the termination status returned to the host environment is undefined.	(until C99)
If the return type of the main function is not compatible with `int` (e.g. `void main(void)`), the value returned to the host environment is unspecified. If the return type is compatible with `int` and control reaches the terminating `}`, the value returned to the environment is the same as if executing `return 0;`.	(since C99)

Example

Demonstrates how to inform a program about where to find its input and where to write its results. Invocation: ./a.out indatafile outdatafile

Run this code

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("argc = %d\n", argc);
    for (int ndx = 0; ndx != argc; ++ndx)
        printf("argv[%d] --> %s\n", ndx, argv[ndx]);
    printf("argv[argc] = %p\n", (void*)argv[argc]);
}
```

Possible output:

```
argc = 3
argv[0] --> ./a.out
argv[1] --> indatafile
argv[2] --> outdatafile
argv[argc] = (nil)
```

References

- C23 standard (ISO/IEC 9899:2023):
  - 5.1.2.2.1 Program startup (p: TBD)
- C17 standard (ISO/IEC 9899:2018):
  - 5.1.2.2.1 Program startup (p: 10-11)
- C11 standard (ISO/IEC 9899:2011):
  - 5.1.2.2.1 Program startup (p: 13)
- C99 standard (ISO/IEC 9899:1999):
  - 5.1.2.2.1 Program startup (p: 12)
- C89/C90 standard (ISO/IEC 9899:1990):
  - 5.1.2.2 Hosted environment

See also

C++ documentation for **`main` function**

Retrieved from "https://en.cppreference.com/mwiki/index.php?title=c/language/main_function&oldid=154725"