What is the proper declaration of main?

What is the proper signature of the `main` function in C++? What is the correct return type, and what does it mean to return a value from `main` ? What are the allowed parameter types, and what are their meanings?

Is this system-specific? Have those rules changed over time? What happens if I violate them?

c++ main c++-faq

edited Nov 18 '10 at 17:15

James McNellis
249k 47 717 853

asked Nov 17 '10 at 17:12

fredoverflow
133k 47 259 528

This is very closely related to, or a duplicate of, What should `main` return in C and C++. – Jonathan Leffler May 17 '14 at 20:02

@JonathanLeffler No kidding... it was added to the list of duplicates in revision 6 about 8 months ago. – fredoverflow May 18 '14 at 10:18

4 Answers

The `main` function must be declared as a non-member function in the global namespace. This means that it cannot be a static or non-static member function of a class, nor can it be placed in a namespace (even the unnamed namespace).

The name `main` is not reserved in C++ except as a function in the global namespace. You are free to declare other entities named `main` , including among other things, classes, variables, enumerations, member functions, and non-member functions not in the global namespace.

You can declare a function named `main` as a member function or in a namespace, but such a function would not be the `main` function that designates where the program starts.

The `main` function cannot be declared as `static` or `inline` . It also cannot be overloaded; there can be only one function named `main` in the global namespace.

The `main` function cannot be used in your program: you are not allowed to call the `main` function from anywhere in your code, nor are you allowed to take its address.

The return type of `main` must be `int` . No other return type is allowed (this rule is in bold because it is very common to see incorrect programs that declare `main` with a return type of `void` ; this is probably the most frequently violated rule concerning the `main` function).

There are two declarations of `main` that must be allowed:

```
int main()              // (1)
int main(int, char*[])  // (2)
```

In **(1)**, there are no parameters.

In **(2)**, there are two parameters and they are conventionally named `argc` and `argv` , respectively. `argv` is a pointer to an array of C strings representing the arguments to the program. `argc` is the number of arguments in the `argv` array.

Usually, `argv[0]` contains the name of the program, but this is not always the case. `argv[argc]` is guaranteed to be a null pointer.

Note that since an array type argument (like `char*[]`) is really just a pointer type argument in disguise, the following two are both valid ways to write **(2)** and they both mean exactly the same thing:

```
int main(int argc, char* argv[])
int main(int argc, char** argv)
```

Some implementations may allow other types and numbers of parameters; you'd have to check the documentation of your implementation to see what it supports.

`main()` is expected to return zero to indicate success and non-zero to indicate failure. You are not required to explicitly write a `return` statement in `main()`: if you let `main()` return without an explicit `return` statement, it's the same as if you had written `return 0;`. The following two `main()` functions have the same behavior:

```
int main() { }
int main() { return 0; }
```

There are two macros, `EXIT_SUCCESS` and `EXIT_FAILURE`, defined in `<cstdlib>` that can also be returned from `main()` to indicate success and failure, respectively.

The value returned by `main()` is passed to the `exit()` function, which terminates the program.

Note that all of this applies only when compiling for a hosted environment (informally, an environment where you have a full standard library and there's an OS running your program). It is also possible to compile a C++ program for a freestanding environment (for example, some types of embedded systems), in which case startup and termination are wholly implementation-defined and a `main()` function may not even be required. If you're writing C++ for a modern desktop OS, though, you're compiling for a hosted environment.

edited Nov 23 at 18:24

answered Nov 17 '10 at 17:19

James McNellis
249k 47 717 853

1 IIRC the only guaranteed return values are 0, EXIT_SUCCESS (same effect as 0), and EXIT_FAILURE. EDIT: Ah, OK, other non-zero status values may be returned, but with implementation-defined meaning. Only EXIT_FAILURE is guaranteed to be interpreted in some way as a failure value. – Derrick Turk Nov 17 '10 at 17:24

1 @Synetech: The question asks in its first sentence, "What is the proper signature of the main function in C++?" and the question is tagged both [c++] and [c++-faq]. I can't help it if Java or C# users (or anyone else) are still confused. C# requires `Main` to be a static member function because it doesn't even have nonmember functions. Even C89 requires `main` to return `int`. I am not sufficiently familiar with K&R C to know its exact rules, but I would guess it also requires `main` to return `int` since `main` with no return type was somewhat common and no type = implicit `int` in K&R. – James McNellis Dec 21 '10 at 20:30

2 @Suhail: Because the language standard says the return type shall be `int`. – James McNellis Jun 15 '11 at 16:35

1 @Suhail: Yes. Your code will not be correct C++ and many compilers will reject your code. – James McNellis Jun 15 '11 at 18:37

2 @Suhail: Visual C++ permits a `void` return type as a language extension. Compilers that do not permit it include GCC and Comeau. – James McNellis Jun 15 '11 at 21:08

From Standard docs., *3.6.1.2 Main Function*,

> **It shall have a return type of type int, but otherwise its type is implementation-defined.** All implementations shall allow both of the following definitions of main:

`int main() { / ... / }` and `int main(int argc, char* argv[]) { / ... / }`

> In the latter form `argc` shall be the **number of arguments passed to the program** from the environment in which the program is run.If argc is nonzero **these arguments shall be supplied in argv[0] through argv[argc-1] as pointers to the initial characters of null-terminated multibyte strings**.....

Hope that helps..

answered Nov 18 '10 at 5:21

liaK
7,634 7 30 60

1 is there any specific reason as to why the return type of `main` should be `int` ? – Suhail Gupta Jun 15 '11 at 12:01

The two valid mains are int main() and int main(int, char*[]) Any thing else may or may not compile. If main doesn't explicitly return a value 0 is implicitly returned.

answered Nov 17 '10 at 17:49

stonemetal
5,384 14 27

I have never seen code not getting compiled when i mention the return type of `main` to be void. **Is there any specific reason that return type of main should be int ?** – Suhail Gupta Jun 15 '11 at 12:03

3 The language specification says main must have a return type of int. Any other return type allowed by your compiler is a compiler specific enhancement. Basically using void means you are programming in a language similar to but not C++. – stonemetal Jun 15 '11 at 23:33

1 The reason the standard requires an `int` as the return type of `main` is that this value is handed to the shell as the program's exit code, and `sh` expects an `int` . – uckelman Jul 12 '13 at 11:55

Details on return values and their meaning

Per 3.6.1 (`[basic.start.main]`):

> A return statement in `main` has the effect of leaving the `main` function (destroying any objects with automatic storage duration) and calling `std::exit` with the return value as the argument. If control reaches the end of `main` without encountering a `return` statement, the effect is that of executing
>
> ```
> return 0;
> ```

The behavior of `std::exit` is detailed in section 18.5 (`[support.start.term]`), and describes the status code:

> Finally, control is returned to the host environment. If status is zero or `EXIT_SUCCESS` , an implementation-defined form of the status successful termination is returned. If status is `EXIT_FAILURE` , an implementation-defined form of the status unsuccessful termination is returned. Otherwise the status returned is implementation-defined.

answered Jan 2 '12 at 2:46

Ben Voigt
206k 21 242 459