Introduction

B++ has its origins in BOB, an approach to a minimalist object-oriented (and C++-like) language developed by David Michael Betz and published in 1991 in Dr. Dobbs Journal.

In order to make the language suitable for practical use, some additions to the language scope itself, in particular memory management, as well as the provision of a number of elementary predefined functions were essential. The result was called BOB+ and was essentially an extension of BOB, which was largely based on the original code. The focus of the development of BOB+ was initially on minimally equipped MS-DOS-based systems. That's why special emphasis was placed on minimal space requirements - both on the data storage medium and in the main memory.

B++ is intended primarily as an easily integrated scripting language for custom extension/control of applications, for quickly creating smaller utility programs and for prototyping.
Achieving this goal required a completely new internal architecture, which also entailed a complete reimplementation.

About B++

B++ is a hybrid language in several ways.

According to its concept, it is initially a procedural language with object-oriented extensions. Functors and anonymous functions, which are also part of B++, allow the use of functional paradigms.

B++ is neither a real compiler nor a real interpreter. The source code of a program is first translated into bytecode, which is then interpreted in a virtual machine. This allows very easy porting of B++ to different target platforms and the execution of existing B++ programs on any supported platform (as long as no platform specific API functions are used directly).

As usual for scripting languages, B++ allows you to work with dynamically typed variables. Static typing is available too, whereby the type checking is carried out partly by the compiler and partly by the runtime system. Both forms can also be used mixed.

Syntactically (and to a certain extent semantically) B++ borrows heavily from C/C++, with some elements from other scripting languages, particularly JavaScript.

The name "B++" refers to its similarity to C++ and its "ancestor" BOB+.

Features

B++ is a reletively small language with a total of 36 keywords, and the basics of it an be learned quickly and easily. But it is also a very powerful and expressive language that supports object orientation and modularization as well as access to system functions.

The following (incomplete) list includes some essential language features of B++, which are presented in more detail below.

- Support for floating point arithmetic (float and double data types) on all platforms, via emulation if necessary
- explicitly usable integer data types from 8 to 64 bits wide, each signed and unsigned, with 32 bits being the standard size
- Integrated data types for strings, buffers, vectors (dynamic arrays) and dictionaries (associative containers)
- integer numeric literals in decimal, binary, octal or hexadecimal notation
- Literal representations for vectors and dictionaries
- Function and method pointers
- anonymous functions and methods
- asymmetric coroutines
- variable parameter lists and default parameters for functions
- polymorphic classes, automatic destructors and initialized static member variables
- partial classes
- custom operators for objects
- Type information at runtime (RTTI)
- Access to system functions
- Integrated serialization of any data
- Use of precompiled bytecode libraries
- dynamic loading of libraries at runtime
- Either explicit memory management or automatic memory management with reference counting
- optional static typing

There are also a variety of predefined functions that fulfill the tasks of a standard library, as well as predefined classes to support regular expressions and to emulate C structures.

The B++ distribution

B++ is currently available for the following platforms:

- MS-DOS from version 3.01: ZIP archive with the binaries for the DOS version and some examples
- Windows desktop 32 bit (Vista, Win7 .. 11) : MSI Installer
- Windows desktop 64 bit (Vista, Win 7 .. 11) : MSI Installer
- Linux (source code only)

  The B++ sources also contain project files for Borland C++ (DOS), Visual Studio 2005 and 2022 (Windows Desktop), CodeLite (Linux) and Embedded Visual Tools 3.0 (Windows CE, deprecated).

  The Windows versions also come with a simple IDE whose task, in addition to making it easier to create B++ programs, is to show how B++ can be embedded in other applications.

The core of all distributions is the **bpplib** library.
It contains the virtual machine with all necessary additional components (compiler, interpreter, predefined functions, etc.). It is available as a dynamic library (DLL) in the Windows versions and in the DOS version as a static C++ library (LIB). For Linux it can be created either as a dynamic or static library.

There is also a command line version **bp2.exe** for all platforms, which is basically a minimal application of the library with a command line interface. It allows the direct use of B++ from a console or the integration of B++ programs in command scripts. There is a modified version of the command line version, **bp2r.exe**, which can be linked to a B++ bytecode module to form an independently executable program. In the Windows desktop versions there is an additional variant **bp2rw.exe**, which differs from bp2r.exe in that it does not contain a console. It is intended for creating 'typical' Windows applications.
In order to be able to use the different versions of Windows in parallel, specific suffixes are appended to the names of the executable files, where **u** stands for a UNICODE variant and **_64** for a 64-bit variant.
For the DOS version there is a variant with the suffix **_386**, which requires at least an 80386 processor. It can use its 32-bit register set and is therefore faster than the standard version, which runs on an 8088 processor too. Both variants use an existing numerical coprocessor or emulation (if no such processor is available) for floating point operations.
The Windows versions include an extension library **bp2class**, which is primarily intended for demonstrating the implementation of extensions. It contains some general classes (string, container, emulation of structures) as well as basic classes for Windows GUI programming. Building on this, the core of a Windows GUI framework is provided as B++ source code. An example that implements a Notepad clone in B++ serves to demonstrate how to use this.

Environment variables

B++ can load extensions in the form of dynamic libraries (usually with the .dll extension) at runtime. The search for such libraries is carried out using the search path1 specified in the PATH environment variable.

The search path specified in the BPPINC environment variable is used to search for source files that are included using the #include processing instruction.

The BPPLIB environment variable serves as a search path for precompiled modules to load. As in PATH, several paths can be stored in BPPINC and BPPLIB, each separated by a semicolon (;).

Using B++ from command line

B++ is started from the command line by calling bp2.exe (or its variants). To simplify calls from anywhere, the path to this file (or files) should be included in the PATH environment variable.
The following syntax then applies to the actual call:

```
bp2 [-i][-d][-l][-v][-t][-c][-e][-E][-L rtname] [-Dsymbol[=value][..]]
[sourcefile [..]] [-r objfile [..]] [-o outfile] [# userarg [..]]
```

The individual options have the following meaning:

- -i (info) Displays copyright and function names.
- -d (debug) Output of the generated bytecode when compiling.
- -l (line info) Source file names and line numbers are included in the bytecode for debugging purposes.
- -v (variables) Names of local variables are written into bytecode for debugging purposes.
- -t (trace) Traces bytecode when programm is executed.
- -c (compile) The specified sources are not executed, but translated into bytecode only.
- -e (make exe) Creates an independently executable program (sources given are translated and the result is linked to the standard runtime module)
- -E (make exe) Creates a standalone program without a console (sources given will be translated and the result with the Standard runtime module without console bound).
  The option only exists in the Windows desktop versions.
- -L (link exe) Creates an independently executable program (specified sources are translated and the result is linked to with the Runtime module specified in `rtname`.
- -D (define symbol) A symbol `symbol` is defined for the subsequent translation, optionally with the value `value`. The symbol defined is a *named literal* of `string`, if a value was specified, otherwise it has the value `null`.
- -r (read / run) Reads (and executes if needed) precompiled bytecode modules
- -o (output) Name of the file to be output. This option is evaluated only, if c, e, E or -L is set too. If the information is missing, the default value out.bpm or out.exe is used.

Source files and modules to be read can be specified without file extensions. In such cases, the extensions .bpp or .bpm are implicitly appended.

If multiple source files are specified, they will be translated into bytecode in the specified order before actual program execution begins. The precompiled modules to be loaded are read *before* compiling the sources (and also in the order specified).

B++ also allows specifying a list of user arguments that are passed on to the program to be executed.
This list is introduced by a hash symbol (#) and forms the end of the command line. Within a B++ program, the entire or the user argument list can be accessed using the predefined functions **getargs()** or **getusrargs()**.
The complete argument list is also passed to the entry function main when called from the command line, so that access to the arguments is also possible using the functions **argcnt()**, arg(i) or **argv()**.
Unlike BOB+, B++ uses a dynamic stack whose size is limited only by available memory. An explicit definition of size using an environment variable is therefore neither necessary nor possible.

B++ Reference © 2005-2026 by R.-Erik Ebert

All parts of B++ are provided under terms and conditions of MIT license.