Harbour (compilador)

O **Harbour** é um [compilador](https://pt.wikipedia.org/wiki/Compilador) moderno e rápido de [software livre](https://pt.wikipedia.org/wiki/Software_livre) para a [linguagem](https://pt.wikipedia.org/wiki/Linguagem) [xBase](https://pt.wikipedia.org/wiki/XBase) (a linguagem que é implementada pelo compilador [Clipper](https://pt.wikipedia.org/wiki/Clipper_(linguagem_de_programa%C3%A7%C3%A3o))).

O Harbour é um compilador multiplataforma e sabe-se que compila e executa em todas elas sem alterações em seu código ou banco de dados.

A principal diferença do Harbour para outros compiladores [dBase](https://pt.wikipedia.org/wiki/DBase) é que ele é um software livre. Mas o Harbour é focado em prover às comunidades Clipper e [xBase](https://pt.wikipedia.org/wiki/XBase) um compilador moderno e portável, possuindo extensões que o tornam um dos mais modernos e completos ambientes de desenvolvimento de aplicações do mercado, em nada lembrando o obsoleto Clipper, exceto pela sua sintaxe e semântica básica, além de seu banco de dados preferencial baseado em tabelas [DBF](https://pt.wikipedia.org/wiki/DBF).

A licença do Harbour[[1]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-1) é similar a [LGPL](https://pt.wikipedia.org/wiki/LGPL) com a exceção de suportar aplicações comerciais para serem construídas e distribuídas através do Harbour.

História

É uma [linguagem](https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o) derivada, de linhagem antiga. Originou-se do compilador Clipper 5, que é uma evolução bastante diferente do Clipper Summer 87, que derivou-se da linguagem encontrada no SGBD dBase III for MS-DOS, que por sua vez, é uma migração do dBase II que rodava no [CP/M](https://pt.wikipedia.org/wiki/CP/M). As raízes da linguagem, portanto, datam das primeiras gerações de linguagens de programação.

A ideia de se criar uma alternativa ao Clipper que já era defunto, circulava pelas discussões em comp.lang.clipper. Em 1999 o espanhol responsável pela conhecida biblioteca de interface gráfica para Windows Fivewin, Antonio Linares iniciou um [tópico](https://groups.google.com/group/comp.lang.clipper/msg/5b4996fdf806504f) para discutir algumas ideais sobre o uso dos softwares [Lex](https://pt.wikipedia.org/wiki/Lex) and [Yacc](https://pt.wikipedia.org/wiki/Yacc) para implementar um compilador semelhante ao Clipper.

"Velejando com o barco [clipper](https://pt.wikipedia.org/wiki/Clipper_(veleiro)) para um porto seguro" foi a frase que serviu de base do nome. Clipper é um tipo de barco. Harbour é sinônimo de porto(onde barcos atracam), em Inglês, então Harbour é o porto seguro para a linguagem Clipper.

Em 2009, o Harbour recebeu uma reformulação geral em sua arquitetura promovida principalmente por Viktor Szakáts e Przemyslaw Czerpak, os principais desenvolvedores do Harbour no momento. Esta reformulação coloca o Harbour entre os melhores compiladores existentes no mercado mesmo quando se compara com outras linguagens.

Público alvo

O compilador Harbour encontra seus principais usuários nos [programadores](https://pt.wikipedia.org/wiki/Programador) Clipper, que já possuem grande quantidade de código escrito e precisam evoluir junto com as constantes inovações do mundo da informática, e também do surgimento de novas necessidades dentro das empresas e instituições onde os sistemas estão implantados. Embora seja uma poderosa linguagem de propósito geral, ela é primariamente utilizada para construir aplicativos de negócios e onde se utilizam grande volumes de dados. Até hoje, os principais fabricantes de equipamentos para automação comercial precisam disponibilizar bibliotecas de acesso aos seus *hardwares* para os programadores xBase interagirem, se desejam ter sucesso no mercado.

Clipper pode ser considerada uma linguagem obsoleta, inclusive por ter inúmeros *bugs* sem solução desde meados da década de 90. Já o Harbour tem posição oposta, sem deixar de lado a compatibilidade com o estilo Clipper 5.x, versões anteriores como Summer'87 e ainda colocando compatibilidade com outros dialetos, permite uma série de inovações na linguagem modernizando-a e colocando no mesmo patamar das linguagem que possuem as últimas novidades. Vale lembrar que muitas das funcionalidades glorificadas hoje em dia em linguagens modernas já estavam presentes no Clipper há mais de 20 anos atrás de uma forma ou de outra.

Características

* [Software livre](https://pt.wikipedia.org/wiki/Software_livre) e multiplataforma ( ([DOS](https://pt.wikipedia.org/wiki/DOS), [Microsoft Windows](https://pt.wikipedia.org/wiki/Microsoft_Windows) (32, 64), [Linux](https://pt.wikipedia.org/wiki/Linux) (32, 64), [Unix](https://pt.wikipedia.org/wiki/Unix) (32, 64), [BSD](https://pt.wikipedia.org/wiki/Berkeley_Software_Distribution), [Mac OS X](https://pt.wikipedia.org/wiki/Mac_OS_X), [OS/2](https://pt.wikipedia.org/wiki/OS/2), [Windows CE](https://pt.wikipedia.org/wiki/Windows_CE), [Pocket PC](https://pt.wikipedia.org/wiki/Pocket_PC), [Symbian](https://pt.wikipedia.org/wiki/Symbian), [iPhone](https://pt.wikipedia.org/wiki/IPhone), [Android](https://pt.wikipedia.org/wiki/Android), [QNX](https://pt.wikipedia.org/wiki/QNX), [VxWorks](https://pt.wikipedia.org/wiki/VxWorks), [Ecomstation](https://pt.wikipedia.org/wiki/Ecomstation), [BeOS](https://pt.wikipedia.org/wiki/BeOS), etc);
* 100% Compatível com o CA-Clipper 5.2 e 5.3[[2]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-2) e possui camadas de compatibilidade para XBase++, Flagship, FoxPro, Visual Objects e xHarbour;
* O Harbour converte os arquivos com extensão PRG em C, usando depois um compilador C para gerar os arquivos executáveis.
* Suporte Técnico especializado em lista de discussão aberta a todos;
* Sua licença permite a criação de programas tanto de código aberto como de código fechado.
* Pré-processador de código-fonte;
* Compiladores de alta-performance para geração do pCode e utilização de compiladores C para gerar o executável;
* Depurador interativo;
* Interface gráfica (opcional, com implementações diferentes, destacando-se a [Qt](https://pt.wikipedia.org/wiki/Qt) colocando o Harbour em pé de igualdade com qualquer outra linguagem moderna no quesito interface);
* Suporte a diversos modos de console de texto ou gráfico (com os drivers adequados) com recursos melhorados em relação ao Clipper;
* Forte integração com a linguagem C e consequentemente com assembly, melhorando e facilitando o que já era muito bom no Clipper, incluindo uma extensiva API revelando boa parte dos componentes internos da plataforma Harbour.

Harbour pode usar um dos seguintes compiladores C, entre outros: [GCC](https://pt.wikipedia.org/wiki/GNU_Compiler_Collection), [MinGW](https://pt.wikipedia.org/wiki/MinGW), [Clang](https://pt.wikipedia.org/wiki/Clang), [ICC](https://pt.wikipedia.org/wiki/Intel_C%2B%2B_Compiler), [Microsoft Visual C++](https://pt.wikipedia.org/wiki/Microsoft_Visual_C%2B%2B) (6+), [Borland C++](https://pt.wikipedia.org/w/index.php?title=Borland_C%2B%2B&action=edit&redlink=1), [Watcom C](https://pt.wikipedia.org/w/index.php?title=Watcom_C&action=edit&redlink=1), [Pelles C](https://pt.wikipedia.org/w/index.php?title=Pelles_C&action=edit&redlink=1) e [Sun Studio](https://pt.wikipedia.org/w/index.php?title=Sun_Studio_(software)&action=edit&redlink=1).

**Informações Complementares**

O Harbour permite a dinamização de aplicações com arquivos de dados, tornando-as mais fáceis e rápidas que as desenvolvidas em uma linguagem de programação tradicional, como Cobol, Basic ou Pascal ou mesmo as mais modernas como Java, C# ou Delphi. Com uma simples, moderna e eficiente linguagem de programação, permite o encadeamento ordenado e lógico de seus comandos possibilitando rapidamente a definição de programas com alto grau de complexidade e sofisticação, permitindo inclusive interações com outras linguagens como “C” e Assembler, que lhe confere a flexibilidade necessária para a utilização profissional.

**Manipulação de saída**

Harbour faz uso de múltiplas emulações e acessos nativos a terminais gráficos incluindo através de drivers específicos para console e terminais híbridos Console/GUIs.

Exemplos de drivers:

**Nativos**

* GtNul, GtCGI, GtStd, GtPCA, GtOS2, GtWin, GtCrs, GtSLN, GtTRM, GTWvt, GTWvg, GtXWc, GtAlleg, GtGUI, GtCTw, HBQt

**Fornecidos por terceiros como software livre**

* HWGui, MiniGUI, HMGExt, ooHG

**Fornecidos por terceiros comercialmente**

* FiveWin, Xailer

HBQt é uma biblioteca que fornece [bindings](https://pt.wikipedia.org/w/index.php?title=Bindings&action=edit&redlink=1) à biblioteca [Qt](https://pt.wikipedia.org/wiki/Qt) de uma forma inovativa. Com a HBQt qualquer programador Harbour pode criar aplicações com visual profissional e nativo à cada plataforma sem nenhuma alteração no código. A aplicação HBIDE inclusa na distribuição oficial e através do repositório SVN é um exemplo do potencial da HBQt.

**Suporte a banco de dados**

Harbour estende os *Replaceable Database Drivers* (RDDs) do Clipper. Esses RDDs são precursores do ODBC permitindo que várias tecnologias de banco de dados possam ser acessados utilizando a mesma base de código.

O Harbour já oferece nativamente múltiplos RDDs tais como [DBF](https://pt.wikipedia.org/wiki/DBF), DBFNTX, DBFCDX, DBFDBT and DBFFPT. É permitida a utilização de vários RDDs simultaneamente e a criação de novos RDDs lógicos frutos da combinação de outros RDDs já existentes, já que sua arquitetura permite a herança dos RDDs de forma semelhante ao que ocorre com a orientação a objeto. RDDs de terceiros, como RDDSQL, RDDSIX, RMDBFCDX, Advantage Database Server, and Mediator exemplificam algumas funcionalidades da tecnologia do RDD. A implementação do DBFNTX tem quase toda funcionalidade existente no DBFCDX (típica do FoxPro) and RDDSIX (através da biblioteca SixDrive). NETIO (nativo do Harbour) e o LetoDB[[3]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-3) fornecem acesso remoto à DBFs através do protocolo [TCP](https://pt.wikipedia.org/wiki/TCP), evitando os problemas de compartilhamento de arquivos encontrados em conexões [SMB](https://pt.wikipedia.org/wiki/SMB) normalmente adotadas em redes locais.

Harbour também oferece suporte ao [ODBC](https://pt.wikipedia.org/wiki/ODBC) com uma sintaxe [OOP](https://pt.wikipedia.org/w/index.php?title=OOP&action=edit&redlink=1), e suporte ao [ADO](https://pt.wikipedia.org/wiki/ActiveX_Data_Objects) através da tecnologia [OLE](https://pt.wikipedia.org/wiki/OLE_Automation). [MySQL](https://pt.wikipedia.org/wiki/MySQL), [PostgreSQL](https://pt.wikipedia.org/wiki/PostgreSQL), [SQLite](https://pt.wikipedia.org/wiki/SQLite), [Firebird](https://pt.wikipedia.org/wiki/Firebird_(servidor_de_base_de_dados)), [Sybase](https://pt.wikipedia.org/wiki/Sybase),[DB2](https://pt.wikipedia.org/wiki/DB2), [Oracle](https://pt.wikipedia.org/wiki/Oracle_(banco_de_dados)), [SQLServer](https://pt.wikipedia.org/wiki/SQLServer) são exemplos de bancos de dados que o Harbour pode interagir nativamente através de suas próprias combinações.

Tecnologias xBase são frequentemente confundidas com softwares [SGDB](https://pt.wikipedia.org/wiki/SGDB). Embora isso seja verdade, essas tecnologias são mais que simples sistemas de banco de dados ao mesmo tempo que uma linguagem baseada em xBase usando apenas o formato DBF não pode fornecer todo o modelo que um SGDB real pode alcançar.

**Extensões**

* [Internet](https://pt.wikipedia.org/wiki/Internet) (http, pop, smtp, CGI, TCP, UDP, RPC), através de [sockets](https://pt.wikipedia.org/w/index.php?title=Sockets&action=edit&redlink=1), bibliotecas TIP, TPathy e [cURL](https://pt.wikipedia.org/wiki/CURL), permitindo a criação de páginas dinâmicas para a web ou mesmo criar seus próprios servidores;
* Criptografia [OpenSSL](https://pt.wikipedia.org/wiki/OpenSSL)
* [Blat](https://pt.wikipedia.org/w/index.php?title=Blat_(software)&action=edit&redlink=1), um software para envio de e-mail;
* [FreeImage](https://pt.wikipedia.org/w/index.php?title=FreeImage&action=edit&redlink=1) e [GD](https://pt.wikipedia.org/wiki/GD) para manipulação de imagens;
* Manipulação de [expressões regulares](https://pt.wikipedia.org/wiki/Express%C3%B5es_regulares) estilo Perl através da [PCRE](https://pt.wikipedia.org/wiki/PCRE);
* Compactação e descompactação de dados através do HbZip ([zlib](https://pt.wikipedia.org/wiki/Zlib) and [bzip2](https://pt.wikipedia.org/wiki/Bzip2))
* [XML](https://pt.wikipedia.org/wiki/XML);
* Implementação própria das bibliotecas CA-Tools e NanFor
* [Unicode](https://pt.wikipedia.org/wiki/Unicode)
* [Multi-Threading](https://pt.wikipedia.org/w/index.php?title=Multi-Threading&action=edit&redlink=1) nativo;
* Suporte a execuções em Background;
* Interface Gráfica para Windows e Linux, OS/2, Mac e outros com várias tecnologias como a Qt da Nokia, base do KDE e [Cairo](https://pt.wikipedia.org/w/index.php?title=Cairo_(gr%C3%A1fico)&action=edit&redlink=1);
* OLE cliente e servidor;
* ADO cliente e servidor;
* OOP seguindo a sintaxe básica da tradicional Class(y);
* Novos tipos de dados, como Hash, Symbol e Pointer, além de sintaxe natural para literal de data (Ex.: 0d20100401);
* Novas sintaxes SWITCH, FOR EACH, etc.;
* Exceções estruturadas ou centralizadas em objeto específico
* Acesso a estruturas C diretamente do código xBase;
* Módulos de carregamento dinâmico (HRB) que funcionam semlhante a uma DLL, mas voltada para integração de um pCode auxiliar à [VM](https://pt.wikipedia.org/wiki/M%C3%A1quina_virtual) principal

Filosofia de Programação

Harbour tem a proposta de ser extremamente portável mas ao contrário de Java que pretende ser *write once, run anywhere* (escreva uma vez, rode em qualquer lugar), Harbour procura ser *write once, compile anywhere* (escreva uma vez, compile em qualquer lugar). Uma vez que o mesmo compilador esteja disponível para todas as plataformas já citadas, não há necessidade de produzir códigos diferentes para cada plataforma, exceto para características exclusivas de um sistema operacional e mesmo assim Harbour sempre procura fornecer soluções portáveis. Compilação cruzada entre plataformas também é oferecida através do compilador [MinGW](https://pt.wikipedia.org/wiki/MinGW)32.

Sob Microsoft Windows, Harbour é mais estável ao passo que também é menos documentado que seu predecessor Clipper, mas a capacidade multiplataforma é mais transparente, personalizável e é portável no sentido de mobilidade, podendo rodar a partir de um dispositivo USB com memória flash.

Sob Linux e Windows Mobile, um código original Clipper pode ser compilado com pouquíssimas adaptações. A maioria dos softwares escritos para compilar com XBase++, Flagship, FoxPro, xHarbour e outros dialetos podem ser compilados com alguma adaptação. Em 2010, muitos esforços estão sendo feitos para facilitar a transição de todos os dialetos [xBase](https://pt.wikipedia.org/wiki/XBase) para o Harbour.

Qualquer linguagem seguindo o padrão xBase foi projetada para oferecer um grande ganho de produtividade no desenvolvimento de aplicações de negócios e de acesso intensivo de bases de dados, utilizando, algumas vezes, técnicas questionáveis por programadores de outras linguagens, mas que funcionam muito bem para a finalidade empregada neste tipo de aplicação. Harbour não é uma exceção.

Paradigmas de programação

**Paradigma Procedural**

Assim como o Clipper é pertencente ao Paradigma procedural (como Pascal, C, Ada, Cobol, Fortran), Harbour utiliza fundamentalmente este paradigma de forma estruturada, seja em sua forma [imperativa](https://pt.wikipedia.org/w/index.php?title=Imperativa&action=edit&redlink=1) ou [declarativa](https://pt.wikipedia.org/w/index.php?title=Declarativa&action=edit&redlink=1).

**Paradigma Orientado a Objeto**

Engana-se quem pensa que o Harbour é apenas procedural. Desde o Clipper 5.x diversos conceitos avançados, muitos derivados direta ou indiretamente do Smalltalk, portanto ela já permitia a orientação a objeto quando isso ainda não era "moda" e que agora chega ao seu pico no Harbour. Ao contrário do Clipper 5 e suas versões seguintes que contavam com umas classes para consumo por parte do programador, o Harbour fornece, opcionalmente, suporte nativo e completo ao paradigma, incluindo uma forma inovadora de implementação de múltipla herança. Segue a sintaxe básica da tradicional criada pela biblioteca Class(y) através do pré-procesador. Há algumas adaptações à sintaxe para facilitar os programadores acostumados com as bibliotecas [Fivewin](https://pt.wikipedia.org/wiki/Fivewin), Clip4Win e TopClass que implementam sintaxe própria.

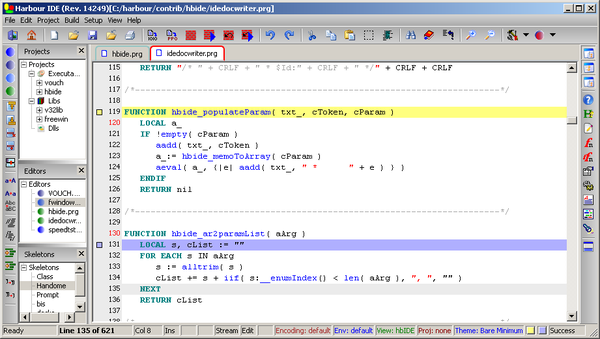
**Paradigma Funcional**

Objetivo alcançado principalmente com o uso de CodeBlocks, uma espécie de [*clausura*](https://pt.wikipedia.org/wiki/Clausura_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)) encontrada em outras linguagens implementada de uma forma peculiar.

**Paradigma de Metaprogramação**

O conceito de pré-processamento do Clipper foi quase revolucionário para a época, onde a macro substituição de texto tão conhecida pelos programadores de C e C++ foi evoluída por um analisador léxico simplificado, tornando a linguagem genérica e incluindo elementos de meta programação.

Sintaxe e semântica

[](https://pt.wikipedia.org/wiki/Ficheiro:Harbour_Sample_Code.png)

Código Harbour na HBIDE.

Harbour, assim como qualquer linguagem xBase não é sensível à caixa de escrita, portanto escrever em maiúscula ou minúscula só é relevante para a organização de cada programador, assim como a linguagem permite opcionalmente aceitar palavras chave escritas apenas com seus 4 primeiros caracteres.

**Sistema de tipos**

Harbour possui uma tipagem dinâmica, porém surpreendentemente forte. O seguinte trecho de código passa pelo compilador, mas dá erro na execução:

LOCAL a,b

a = 1

b = 2

b = "x"

? a + b

O Clipper 5 introduziu a diretiva "local", que permite declarar as variáveis no início da função ou procedimento. Foi um passo em direção à tipagem estática; o plano da Nantucket que era tornar o Clipper mais parecido com C ao longo de sua evolução pode estar sendo retomado agora com o Harbour. O dialeto Visual Objects introduziu a especificação do tipo opcionalmente para que o compilador possa forçar a verificação do tipo utilizado. Harbour segue este padrão, podendo produzir erros em tempo de compilação.

LOCAL a AS NUMERIC, b AS CHARACTER

a = 1

b = 2 //erro em tempo de compilação se a opção de verificação estática de tipos estiver ligada

b = "x"

? a + b //também gerará erro em tempo de compilação

**Tipos**

* **NUMERIC:** Aceita apenas números com ou sem casas decimais (decimals). O ponto da casa decimal conta no tamanho (width). Não há distinção entre números inteiros ou decimais. Exemplo:

nTotal := 0

nCampo := 125

nValor := 25.75

* **CHARACTER:** Aceita letras e números. Tamanho máximo de 256 caracteres. Exemplo:

cNome := SPACE(35)

cCampo := "JOÃO DA SILVA"

cHello := 'Hello'

* **DATE:** Aceita apenas datas. Considere o uso do comando SET DATE BRITISH para definir o padrão de data igual ao brasileiro dd/mm/aa. Considere também o uso do comando SET EPOCH TO 1980 para resolver o "Bug do Milênio", veja mais em Know-how - Y2K Bug do Ano 2000. Exemplo:

dData := CTOD("") //notação antiga aceita pelas maioria dos dialetos xBase

dHoje := DATE()

dDataFinal := 0d20100420 //Notação moderna recomendada pelo Harbour para o dia 20 de abrial de 2010

* **LOGIC:** Aceita apenas Verdadeiro (.T.) ou Falso (.F.). Exemplo:

lCompleto := **.T.**

lErro := **.F.**

* **MEMO:** Aceita letras e números, é como um tipo de campo Caracter aceitando tamanhos variáveis inclusive acima de 256 caracteres. Usado geralmente para armazenar "Observação" ou outras informações que precisem ser detalhadas. Ao adicionar um campo Memo no seu DBF, é criado um espelho dele com a extensão .DBT. (Se o RDD usado for o nativo do Clipper DBFNTX). Normalmente um campo Memo é editado com a função MEMOEDIT(). O tipo Memo é usado apenas em banco de dados DBF e seu conteúdo é transposto para a memória como um tipo Character normal. Exemplo:

mObs := MEMOREAD("NOTAS.TXT")''

* **NIL:** Nil tem como único valor o próprio Nil e indica que o tipo é indefinido. Exemplo:

xVar := Nil

* **ARRAY:** Também chamado de matriz ou vetor. São como vários campos agrupados em uma lista. É uma estrutura de dados que contém uma série de dados ordenados, chamados "elementos". Os elementos são referenciados por número ordinal, iniciando em 1. Exemplo:

aVetor := { "texto", 1, { "Vetor aninhado aqui" }, **.T.**, ResultadoDeFuncao(), @PonteiroDefuncao() }

* **HASH** Hashes podem utilizar qualquer tipo de dado como chave para qualquer elemento. Hashes e Arrays podem conter qualquer tipo de dado como valor de cada elemento, incluindo hashes e arrays. Exemplo:

hTabela := '{ "Nome" => "João", 1 => "Chave Numérica", { "Aninhamento aqui" => "Valor" } }''

* **POINTER** Cria uma referência para uma função ou procedimento já existente. É uma forma mais prática e mais rápida de se utiliizar um bloco de código que apenas chama uma função específica.

pNome := @NomeDaFuncao()

pFuncao := @Int()

* **CODEBLOCK:** É um tipo especial de variável que armazena um pedaço de código compilado, também chamado de função sem nome. Funciona de forma muito semelhante às funções [lambdas](https://pt.wikipedia.org/w/index.php?title=Lambdas&action=edit&redlink=1) ou [clausuras](https://pt.wikipedia.org/wiki/Clausura_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)). Codeblocks podem ter referências para varáveis definidas em *Procedure* ou *Function* onde eles foram definidos. Tais Codeblocks podem ser retornados como valor ou como um argumento passado por referência, neste caso o bloco de código irá sobreviver à rotina em que ele foi definido e qualquer variável que ele referencia será destacada na memória para preservar sua vida.

Codeblocks podem ser executados por qualquer número de vezes através da função Eval( *BlockExp* ) ou suas variantes como AEval() e DBEval(), por exemplo. Exemplo:

@ 10,10 SAY "CODIGO:" GET WCOD PICT "999999" VALID EVAL( { || WCOD := STRZERO(WCOD,6), **.T.**} )

bCodigo := { | x, y | QOut( x + y ) }

* **OBJECT:** Cria um objeto armazenando uma instância de uma classe prédefinida.

oExemplo := ClasseExemplo():New()

QOut(oExemplo:Propriedade)

Outros tipos podem ser assinalados para tabelas DBF, mantendo compatibilidade com tabelas DBF utilizadas pelo Visual FoxPro e outros dialetos. É comum entre os programadores xBase, a utilização da [notação húngara](https://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o_h%C3%BAngara) para as variáveis e campos.

**Escopo de variáveis**

* *LOCAL*: Visível somente na rotina onde ela é declarada. O valor é perdido quando da saída da rotina.
* *STATIC*: Visível somente na rotina onde ela é declarada. Valor é preservado para chamadas subsequentes da rotina. Se uma variável STATIC é declarada antes de qualquer Procedure/Function/Method ser definido, ela tem escopo de módulo, e é visível de qualquer rotina definida dentro do mesmo arquivo fonte, e também mantém o valor durante toda a execução da aplicação.
* *PRIVATE*: Visível na rotina onde é declarada e todas as rotinas que são chamadas por essa rotina. Uma forma de declaração automática quando nada é declarado e que deve ser evitada em programas modernos que pretendam ser seguros.
* *PUBLIC*: Visível por toda a aplicação durante toda a execução. Seu uso deve ser evitado.

*LOCAL* e *STATIC* são resolvidas em tempo de compilação, e portanto são mais rápidas que variáveis *PRIVATE* e *PUBLIC* que são resolvidas pelas em tempo de execução através de uma tabelas de símbolos.

Pela natureza dinâmica das variáveis *PRIVATE* e *PUBLIC*, elas podem ser criadas e destruídas em tempo de execução, mas exatamente por essa flexibilidade elas são criticas pelo abuso no seu uso.

**Controle de seleção**

**IF...ELSE[IF]...ENDIF** Executa uma estrutura de controle, de acordo com a condição ser verdadeira.

**Exemplo:**

**If** !Empty(cNome)

@ 15,10 Say "Nome: " + cNome

**Else**

@ 15,10 Say "-------------------"

**EndIf**

**SWITCH...CASE...DEFAULT...ENDSWITCH** Executa uma estrutura de controle, de acordo com um padrão constante se enquadrar na variável. Funciona de forma análoga à semântica da linguagem C com o adendo de permitir mais tipos de dados sendo comparados.

**Exemplo:**

Switch xVar

**Case** 1

@ 10,00 Say "Opção 1 escolhida"

**Exit**

**Case** "A"

@ 10,00 Say "Opção A escolhida"

**Exit**

Default

@ 10,00 Say "Nenhuma opção válida escolhida"

EndSwitch

**Laços de repetição**

**FOR...NEXT** Executa uma estrutura de controle, um determinado número de vezes.

**Exemplo:**

**For** I := 1 To 100

@ 15,10 Say "CONTADOR:" + Str(I,3)

**Next**

**For** J := 100 To 500 Step 10

@ 18,05 Say "O VALOR DE J É " + StrZero(J,3)

**Next**

**FOR EACH...NEXT** Executa uma estrutura de controle através de de uma variável escalar ou complexa, como array, hash, objetos e character.

**Exemplo:**

**For** Each I In aVetor

@ 18,05 Say "O VALOR DE I É " + PadL(I, 6, " ")

**Next**

Existem funções para manipular de forma mais especifica a interação. A função *HB\_EnumIndex()* por exemplo pode opcionalmente retornar um índice da interação.

**DO WHILE...ENDDO** Executa uma estrutura de controle enquanto uma condição for verdadeira.

**Exemplo:**

while **.T.**

xnumero:=0

@ 11,10 say "Digite um número"

@ 11,20 get xnumero

**read**

**if** empty(xnumero)

**exit**

**endif**

@ 13,10 say "o número digitado foi"+strzero(xnumero,3)

end

xResp := "S"

**do** while xResp != "N" // # ou <> também são símbolos de diferente

xNome := space(40)

@ 11,10 say "Nome: "

@ 11,25 get xNome

**read**

**if** lastkey()==27 // == exatamente igual

**exit**

**endif**

@ 15,18 say "O nome digitado foi: " + xNome

xResp:=space(01)

@ 20,10 say "Deseja continuar ?"

@ 20,30 get xResp picture "!"

**read**

**enddo**

Em todos os casos é possível utilizar os comandos LOOP e EXIT. Eles servem respectivamente para dar continuidade em um laço de repetição ou sair do laço, cancelando as repetições.

**Controle de exceção**

**BEGIN SEQUENCE...RECOVER...ENDSEQUENCE** Executa uma estrutura de controle enquanto uma condição for verdadeira.

**Exemplo:**

Begin Sequence

QOut( 1 + 2 )

Break

QOut("Nunca será executado")

Recover

QOut("Uma exceção ocorreu")

EndSequence

A estrutura BEGIN SEQUENCE controlar a forma que uma sequência de operações abortará em caso de resultados inesperados, até mesmo em funções ou procedimentos aninhados. Em uma Procedure/Function, pode conter um comando BREAK ou uma função Break() para forçar a interrupção do procedimento em andamento desviando para a cláusula RECOVER ou ao fim da sequência. O comando BREAK pode conter expressões que ajudem o RECOVER manipular o erro de forma mais específica.

Harbour adicionalmente tem o objeto *Error Object* que suporta as propriedades *canDefault*, *canRetry* and *canSubstitute*, que permitem que os manipuladores de erro façam alguma preparações, e então podem proceder operações como *Retry*, *Resume*, ou retornar um valor que substitua a expressão que disparou o erro.

**Criação de rotinas**

**FUNCTION/PROCEDURE...RETURN** Executa uma estrutura de controle enquanto uma condição for verdadeira.

**Exemplo:**

x := Cube( 2 )

**Function** Cube( n )

**Return** n \*\* 3

Procedures e Functions podem utilizar o qualificador *STATIC* para restringir seu uso ao módulo onde ela é definida.

Os qualificadores opcionais *INIT* ou *EXIT* podem ser utilizados para definir uma chamada dessas rotinas antes de iniciar a execução da aplicação ou após encerrar sua execução.

Parâmetros podem ser passados para essas rotinas e as variáveis criadas como parâmetros tem escopo *LOCAL*, podendo ser de qualquer tipo, incluindo referências. Mudanças nas variáveis passadas como parâmetro em procedure/function/method não são refletidas nas variáveis utilizadas na chamada a não ser que explicitamente sejam passadas *por referência* com o prefixo *@*.

Procedure não permite retornar um valor, enquanto que uma função deve retornar uma expressão de qualquer tipo, inclusive Nil que a equipara a uma Procedure.

**Exemplos de comandos básicos**

**@. . . SAY. . . GET** Criar e executar um novo objeto GET (entrada de dados),colocando-o em exibição na tela. **Exemplo:**

@ 15,10 SAY "Código"

xCod:=0

@ 18,10 GET xCod

**read**

**Exemplos de código**

? "Alô Mundo!"

**Use** "cliente" Shared New

**Clear** Screen

**Do** While LastKey() != 27

@ 01, 0 Say "Codigo " Get cliente->codigo Pict "999999" Valid cliente->codigo > 0

@ 03, 0 Say "Nome " Get cliente->nome Valid ! Empty(cliente->nome)

@ 04, 0 Say "Endereco" Get cliente->endereco

**Read**

**Enddo**

Obs: O código acima viola a regra básica do uso de tabelas em modo compartilhado.

Estilo antigo xBase:

Local OPC :=1

**SET** WRAP ON // habilita a rolagem da barra entre os extremos do menu

**SET** MESSAGE TO 23 CENTER // determina a saida de mensagens da linha 23 da tela

**DO** WHILE **.T.**

**CLEAR** // LIMPA A TELA

// cria variáveis para facilitar as coordenadas do menu

L:=8

C:=32

// montar a tela

@ 01,01 TO 24,79 DOUBLE

@ 02,02 TO 04,78

@ 03,01 SAY "SISTEMA ADMINISTRATIVO"

@ 03,60 SAY DATE( )

@ 03,70 SAY TIME( )

// detalha o menu de barras

@ L,C PROMPT "INCLUSÃO" MESSAGE "INCLUSAO DE DADOS"

@ L+1,C PROMPT "ALTERAÇAO" MESSAGE "ALTERAÇAO DE DADOS"

@ L+2,C PROMPT "CONSULTA" MESSAGE "CONSULTA DE DADOS"

@ L+3,C PROMPT "EXCLUSAO" MESSAGE "EXCLUSAO DE DADOS"

@ L+4,C PROMPT "RELATORIOS" MESSAGE "RELATORIOS DO SISTEMA"

@ L+5,C PROMPT "UTILITARIOS" MESSAGE "UTILITARIOS DO SISTEMA"

@ L+6,C PROMPT "F I M" MESSAGE "RETORNO AO DOS"

// executa o menu e **control**a a barra

**MENU** OPC

**DO** CASE // faça os casos

**CASE** OPC = 1

**DO** PROG1

**CASE** OPC = 2

**DO** PROG2

**CASE** OPC = 3

**DO** PROG3

**CASE** OPC = 4

**DO** PROG4

**CASE** OPC = 5

**DO** PROG5

**CASE** OPC = 6

**DO** PROG6

**CASE** OPC = 7

**CANCEL** // cancela a execução do programa

**ENDCASE**

INKEY(0) // aguarda QQ tecla

**ENDDO**

**Exemplos OOP**

#include "hbclass.ch"

**PROCEDURE** Main()

LOCAl oPessoa := Pessoa( "Maria" )

oPessoa:Olhos := "Inválido"

oPessoa:Olhos := "Azul"

Alert( oPessoa:Descreve() )

**RETURN**

CLASS Pessoa

DATA Nome INIT ""

METHOD New() CONSTRUCTOR

ACCESS Olhos IN**LINE** ::pvtOlhos

ASSIGN Olhos( x ) IN**LINE** IIF( ValType( x ) == 'C' **.AND.** x IN "Azul,Castanho,Verde", ::pvtEyes := x, Alert( "Valor Inválido" ) )

// exemplo de método IN**LINE**

INLINE METHOD Descreve()

LOCAL cDescricao

**IF** Empty( ::Nome )

cDescricao := "I have no name yet."

**ELSE**

cDescricao := "My name is: " + ::Name + ";"

**ENDIF**

**IF** ! Empty( ::Olhos )

cDescricao += "a cor dos meus olhos é: " + ::Olhos

**ENDIF**

ENDMETHOD

**PRIVATE**:

DATA pvtOlhos

ENDCLASS

// Definição comum de um método.

METHOD New( cNome ) CLASS Pessoa

 ::Nome := cNome

**RETURN** Self

**Macro compilador**

Uma das mais poderosas funcionalidades das linguagens xBase é o [Macro](https://pt.wikipedia.org/w/index.php?title=Macro_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)&action=edit&redlink=1) operador '&'. A implementação do Harbour permite que uma expressão compilada em tempo de execução posa ser usada como VALUE, isto é, pode ser usada para o lado direito de uma atribuição de valor (rvalue), mas o mais interessante é que uma expressão compilada pode ser usada para resolver o lado esquerdo da atribuição (lvalue).

Adicionalmente, o operador de Macro pode compilar e executar chamadas de função, atribuições completas ou uma lista de argumentos e o resultado de uma macro pode ser usado para resolver qualquer dos contextos citados acima em uma aplicação já compilada. Em outras palavras, qualquer aplicação Harbour pode ser estendida e modificada em tempo de execução para compilar e executar código adicional sob demanda de forma segura.

As mais novas versões do compilador de macros pode compilar qualquer código válido escrito em Harbour inclusive código que precise ser préprocessado antes de compilar, dando uma capacidade reflexiva à linguagem dificilmente encontrada em outras linguagens.

Sintaxe:

&( ... )

O texto da expressão '...' será compilado e executado e o valor resultante dessa execução será o resultado final de toda a expressão.

&SomeId

é a forma curta der &( SomeId ).

&SomeId.postfix

é a forma curta de &( SomeId + "postfix" ).

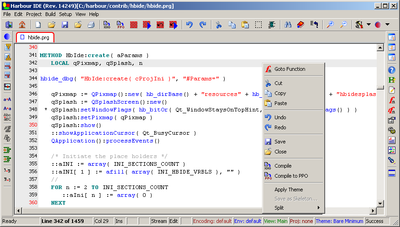
**Préprocessamento**

O préprocessador do Harbour é uma das ferramentas mais poderosas existentes em linguagens de programação, permitindo emular inúmeras características de uma metalinguagem.

Diretivas como #command e #translate permitem alcançar resultados além do que a diretiva #define, comumente encontrada em linguagens como C, consegue, facilitando a criação de nova sintaxe na linguagem. Através dessas diretivas é que o Harbour consegue ser compatível em sintaxe com inúmeros dialetos e também permite pacificar as semânticas diferentes entre eles em uma só sintaxe.

Recursos como este tornam o Harbour uma ferramenta importante quando se fala em plataformas para [DSL](https://pt.wikipedia.org/w/index.php?title=DSL_(Linguagem_de_Dom%C3%ADnio_Espec%C3%ADfico)&action=edit&redlink=1).

Ferramentas

[](https://pt.wikipedia.org/wiki/Ficheiro:HBIDE_Editor.png)

Visual da HBIDE

* HBIDE - Ambiente de desenvolvimento para o Harbour e outros dialetos xBase
* HBMK2 - Construtor muito poderoso estilo [MAKE](https://pt.wikipedia.org/w/index.php?title=MAKE&action=edit&redlink=1)
* HBDoc2 e HBExtern - Gera documentação para o Harbour
* HPPP - Préprocessador, um dos mais poderosos existentes evitando os problemas típicos do equivalente da [linguagem C](https://pt.wikipedia.org/wiki/Linguagem_C) por exemplo
* HBFormat - Formata os códigos fonte escritos em Harbour ou outros dialetos de acordo com determinados critérios
* HBi18n - Ferramenta que facilita a localização de aplicações para outros idiomas
* HBRun - Interpretador de códigos Harbour em linha de comando ou em lote. Através da macro compilação esta ferramenta consegue executar qualquer código válido como se estivesse compilado

Todas essas ferramentas são multiplataforma.

Desenvolvimento

Hoje o desenvolvimento do Harbour está sendo conduzido pelo húngaro Viktor Szakáts com grandes colaborações em vários componentes do núcleo do compilador e componentes auxiliares lideradas pelo polonês Przemyslaw Czerpak. O ambiente de desenvolvimento HBIDE e alguns outros componentes, principalmente a camada HBQt para a cessar os componentes da Qt, são desenvolvidos pelo indiano Pritpal Bedi. Muitos outros participantes da comunidade fazem contribuições no [repositório](https://pt.wikipedia.org/wiki/Reposit%C3%B3rio) [SVN](https://pt.wikipedia.org/wiki/SVN) da [Sourceforge](https://pt.wikipedia.org/wiki/Sourceforge) ou através da lista de discussão.

Em 2010 o desenvolvimento do Harbour se mantém em grande atividade.

Popularidade

Não há uma maneira de avaliar a popularidade de uma linguagem de forma científica, principalmente no caso da popularidade do Harbour que é um dialeto xBase. O índice *TIOBE Programming Community Index* [[4]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-4) de Junho de 2006 apresentou o Microsoft Visual FoxPro, um dos principais dialetos xBase, na 12ª posição no ranking de popularidade de linguagens de programação. Em Julho de 2008, o mesmo índice listava **FoxPro/xBase** na 23ª posição do ranking, passando à 25ª posição em Agosto de 2010. Isso ilustra uma perda de popularidade dentro dos critérios do TIOBE ao mesmo tempo que mostra que dialetos xBase ainda são muito relevantes dentro do cenário mundial da programação, tendo resultados muito mais expressivos que outras linguagens que aparentam estar mais "na moda". De qualquer forma os critérios do TIOBE são bastante questionáveis. Sabe-se empiricamente que usuários xBase não são participativos em grupos pela internet, prejudicando seu posicionamento. Aplicativos desenvolvidos com algum dialeto xBase estão funcionando em milhões de empresas do mundo controlando bilhões de dólares todos os anos.

A listagem do *site* SourceForge listava o Harbour entre os 16 compiladores mais baixados semanalmente e estava na posição 126 no ranqueamento global em Agosto de 2010[[5]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-5).

Comparação com xHarbour

xHarbour é uma bifurcação[[6]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-6) no desenvolvimento do Harbour. xHarbour sempre tomou uma posição mais agressiva na adoação de novas tecnologias, enquanto o Harbour preferiu se concentrar na compatibilidade com o Clipper em um primeiro momento para depois implementar novas funcionalidades. Harbour também se concentrou em suportar uma grande variedade de sistemas operacionais enquanto o xHarbour suporta basicamente o MS Windows e o Linux 32-bit.

Os desenvolvedores do Harbour tentam documentar todos os comportamentos que não são oficialmente documentos pelo Clipper e testar todo código compatível com Clipper para compilar com o Harbour mantendo sua compatibilidade total, incluindo os *bugs* não críticos do Clipper.

Os desenvolvedores do Harbour rejeitam mudanças que quebram a compatibilidade com o Clipper, embora essas rejeições estão sendo revistas agora que o Harbour tem uma arquitetura muito bem implementada e permite extensões à linguagem sem alterar o núcleo do compilador.

Em 2010 Harbour teve um ganho considerável na sua adoação após o lançamento da versão 2.0 que provou ser muito estável, poderosa, flexível e moderna, enquanto o xHarbour tem sido negligenciado pelos seus desenvolvedores e vem perdendo apoio por parte dos seus usuários, como pode ser verificadas em suas listas [[7]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-7)[[8]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-8)[[9]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-9).

Recentemente um texto chamado *xhb-diff.txt*[[10]](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_note-10) disponível na distribuição do Harbour demonstra as inúmeras deficiências do xHarbour em termos de arquitetura que praticamente impossibilitarão a adoção de forma consistente de futuras modernizações na linguagem sem causar mais compromissos do que já carrega.

Referências

* 1. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-1) [*«Licença Harbour»*](https://web.archive.org/web/20120309055958/http:/www.harbour-project.org/license.html)*. Consultado em 7 de abril de 2010. Arquivado do*[*original*](http://www.harbour-project.org/license.html)*em 9 de março de 2012*
  2. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-2) [*«página oficial do Harbour»*](https://web.archive.org/web/20120605041921/http:/www.harbour-project.org/)*. Consultado em 4 de abril de 2010. Arquivado do*[*original*](http://www.harbour-project.org/)*em 5 de junho de 2012*
  3. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-3) [LetoDB](http://sourceforge.net/projects/letodb)
  4. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-4) [TIOBE Programming Community Index](http://www.tiobe.com/tpci.htm)
  5. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-5) [*«SourceForge»*](https://web.archive.org/web/20081118033645/http:/sourceforge.net/softwaremap/)*. Consultado em 10 de abril de 2010. Arquivado do*[*original*](http://sourceforge.net/softwaremap/)*em 18 de novembro de 2008*
  6. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-6) [About xHarbour](http://www.xharbour.org/index.asp?page=about/index)
  7. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-7) [*«estatísticas da lista de discussão Harbour»*](https://web.archive.org/web/20120313164521/http:/gmane.org/info.php?group=gmane.comp.lang.harbour.devel)*. Consultado em 10 de abril de 2010. Arquivado do*[*original*](http://gmane.org/info.php?group=gmane.comp.lang.harbour.devel)*em 13 de março de 2012*
  8. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-8) [*«estatísticas da lista de discussão xHarbour'»*](https://web.archive.org/web/20120313164608/http:/gmane.org/info.php?group=gmane.comp.lang.xharbour.devel)*. Consultado em 10 de abril de 2010. Arquivado do*[*original*](http://gmane.org/info.php?group=gmane.comp.lang.xharbour.devel)*em 13 de março de 2012*
  9. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-9) [*«ohloh.net Activity comparison»*](https://web.archive.org/web/20121012075452/http:/www.ohloh.net/p/compare?metric=Activity&project_0=harbour+project&project_1=xHarbour+Extended+Harbour+Compiler)*. Consultado em 15 de abril de 2010. Arquivado do*[*original*](http://www.ohloh.net/p/compare?metric=Activity&project_0=harbour+project&project_1=xHarbour+Extended+Harbour+Compiler)*em 12 de outubro de 2012*
  10. [↑](https://pt.wikipedia.org/wiki/Harbour_(compilador)#cite_ref-10) [xhb-diff.txt](http://harbour-project.svn.sourceforge.net/viewvc/harbour-project/trunk/harbour/doc/xhb-diff.txt)[*[ligação inativa]*](https://pt.wikipedia.org/wiki/Wikip%C3%A9dia:Liga%C3%A7%C3%A3o_inativa)

Ligações externas

* [Sítio oficial](https://harbour.github.io/) (em inglês)
* [*«hbIDE»*](http://hbide.vouch.info/)*(em inglês). - ambiente de desenvolvimento integrado*
* [*«Clipper On Line»*](http://www.pctoledo.com.br/forum/)*. - fórum sobre Clipper*
* [CA-Clipper Website](https://www.linguagemclipper.com.br/) (em português brasileiro)