Ferramentas xBase

(Sávio Gonçalves – savio_goncalves@yahoo.com.br)

Este documento tem por objetivo ajudar aqueles que estão se iniciando no mundo xBase. Com ele, aquele velho Clippeiro resistente, que não tem mais ânimo para mudar porque acha que terá que estudar muito, ficará surpreso ao descobrir que migrar um sistema seu em Clipper para o xHarbour é a coisa mais fácil deste mundo.

Vamos lá, Clippeiro! Leia este documento, conheça um pouco mais do mundo xBase e siga os passos propostos para instalar o aparato em sua máquina. Após tudo instalado, você jamais vai querer saber do compilador Clipper, mas vai continuar programando nele.

Experimente. Foi assim comigo. Será assim com você.

1 - Conhecendo o mundo xBase:

O termo genérico "xBase" é atribuído a todas as linguagens de programação que derivam do original dBase da Ashton-Tate, como o dBase III Plus, de 1986, e que tem por objetivo trazer a linguagem do dBase (antecessor do Clipper) de 16 bits para o mundo 32/64 bits do Windows, Windows CE, Unix, Linux, Mac OS X, Pocket PC, etc., enfim, tornar a linguagem multi-plataforma.

O termo também serve como referência a tudo o que utiliza os arquivos com extensão .DBF/.DBT e seus índices (.NTX, .NDX, .CDX, .MDX, etc.). Até mesmo utilitários como o DBF Viewer 2000, que é uma versão para Windows do velho DBU, é uma ferramenta xBase.

Também são ferramentas xBase as GUIs (Graphical User Interface), que são bibliotecas gráficas, as IDEs (Integrated Development Environment), que são ambientes de desenvolvimento, e tudo o mais que se refere à herança moderna dos velhos dBase e Clipper.

Várias ferramentas xBase surgiram na tentativa de manter viva a linguagem do antigo dBase/Clipper, trazendo-a para o mundo 32/64 bits. Muitas também fracassaram, caíram em desuso ou nem mesmo emplacaram. Algumas ficaram famosas, como o FlagShip, que é alemão e é pago (www.fship.com).

Mas o grande destaque de toda a família xBase é fruto de uma idéia genial: "The Harbour Project" (www.harbour-project.org), cuja filosofia era a mesma do Linux: criar uma grande comunidade de desenvolvedores e colaboradores ao redor do mundo e disponibilizar uma linguagem free, fervilhante de novas features e extensões e em constante evolução, como se fosse um Clipper turbinado, com novas funções além das originais capazes de, por exemplo, utilizar DLLs, comunicar-se diretamente com a API do Windows, utilizar-se das vantagens do kernel do Linux e, principalmente, contar com centenas de bibliotecas gráficas (GUIs), IDEs e utilitários disponibilizados por desenvolvedores dos quatro cantos do mundo.

"The Harbour Project", conhecido simplesmente por "Harbour", prima pela replicação exata do comportamento do Clipper, fazendo com que suas funções reproduzam exatamente o mesmo resultado seja em qual sistema operacional for. Em segundo plano, o Harbour trata de desenvolver novas funções e extensões para a linguagem.

A partir do Harbour, surge uma nova vertente: o "xHarbour" (www.xharbour.org). O "x" significa "extended". Explicando: como o "The Harbour Project" é open source, um grupo de desenvolvedores resolveu criar um projeto paralelo onde o principal objetivo é uma agressividade maior na criação de novas funções e extensões, sem se preocupar tanto com a total exatidão de comportamento perante às velhas funções do Clipper, o que demanda testes exaustivos em vários sistemas operacionais. Mesmo assim, o xHarbour dá conta de seu recado na hora de converter um sistema em puro Clipper.

Resumindo: o pessoal do Harbour é mais conservador, enquanto o pessoal que desenvolve o xHarbour é mais agressivo na criação de novas funcionalidades.

Posteriormente, surgiu o "xHarbour Builder" (www.xharbour.com), que é a versão comercial (paga) do xHarbour, hoje encontrado nas distribuições Personal, Professional e Enterprise, com suporte e tudo mais.

Apesar de eu ter optado por utilizar o xHarbour "free" (www.xharbour.org) combinado com o compilador Borland C++ 5.5, a GUI HwGUI e a IDE xDevStudio (todos "free"), vale a pena conhecer alguns membros da família xBase:

a) Softwares Comerciais (pagos):

Compiladores:

dBase (dBase Inc. – www.dbase.com)
Clipper (GrafxSoft – www.grafxsoft.com/Clipper.htm)
FoxPro (Microsoft Corp. – www.microsoft.com)
Visual dBase
Visual Objects (Windows 32) (GrafxSoft – www.grafxsoft.com)
Vulcan.NET (GrafxSoft – www.grafxsoft.com)
Visual FoxPro (Microsoft Corp. – www.microsoft.com)
Recital (Recital Corp. – www.recital.com)
Arago
Force
dbFast
dbXL
QuickSilver
xBase++ (Alaska Software – www.alaska-software.com)
Visual FlagShip (Multisoft – www.fship.com)
xHarbour Builder (xHarbour.com Inc. - www.xharbour.com)

GUI (Bibliotecas Gráficas):

FiveWin

IDE (Ambientes de Desenvolvimento):

Xailer

Bibliotecas:

EasyReport

Utilitários:

DBF Viewer 2000 (HiBase Group – www.dbf2002.com)

b) Softwares Free (grátis):

Compiladores:

Clip (Compiler)
Harbour (The Harbour Project – www.harbour-project.org)
xHarbour (xHarbour.org - www.xharbour.org)

GUI (Bibliotecas Gráficas):

HwGUI
MiniGUI
ooHG

IDE (Ambientes de Desenvolvimento):

xDevStudio (www.sqllib.com.br)

Bibliotecas:

SQL LIB (biblioteca para acesso a vários BDs relacionais) (www.sqllib.com.br)

Utilitários:

EG-Expert Guide (Norton Guide for Windows)

2 – Como funciona o xHarbour:

O xHarbour não funciona sozinho. Ele precisa de um compilador C instalado previamente na máquina.

Quanto ao editor, você pode usar qualquer um (até mesmo o EDIT do DOS) ou utilizar uma IDE como a xDevStudio (como aquele ambiente de programação do Delphi) que, além de editor, possui botões para criação, abertura e fechamento de projetos, compilação, debuger, etc., enfim, trabalhando como um gerenciador de projetos.

É recomendável o uso de uma IDE. Do contrário, você terá que aprender a utilizar o HBMake, que gera o script de compilação do seu projeto (makefile), para que você possa compilá-lo no console do DOS (como aqueles .RMK e .LNK do Clipper).

Embora chamado de compilador, o xHarbour, na verdade, é um conversor da linguagem Clipper para C. Seu script de compilação chama o xHarbour que pega arquivos .PRG e gera arquivos .C. Em seguida, chama o compilador C instalado na máquina para que gere o executável a partir dos arquivos .C.

Assim, como produto final, temos um executável de 32 bits em puro C/C++, ainda que com aquela mesma aparência do velho Clipper (modo texto). O executável fica muito mais veloz do que o compilado com o compilador CA-Clipper, de 16 bits.

A única exigência é que seu projeto tenha uma função Main( ) para indicar ao xHarbour quem é o arquivo-fonte principal. Tendo isto e não fazendo uso de bibliotecas de terceiros, ou seja, utilizando apenas funções do CA-Clipper, a migração para o xHarbour é instantânea, sem trabalho algum, com garantia de 100% de compatibilidade.

Se, após migrar de Clipper para xHarbour, você desejar mudar a aparência do sistema para modo gráfico, com janelas como as do Delphi, por exemplo, escolha uma das dezenas de GUIs disponíveis, instale-a e reescreva as telas de seu sistema. Não é uma tarefa tão simples, mas o resultado final é excelente.

Resumindo: para começar, você precisa de apenas 3 coisas: o compilador C (baixe-o em www.xharbour.org), o compilador xHarbour (www.xharbour.org) e uma IDE. Eu recomendo começar com a IDE xDevStudio versão 0.72 Black Edition (www.sqllib.com.br), que gera o script de compilação Build_MyPRG.bat (caso queria compilar diretamente no DOS, por fora da IDE) e ainda o arquivo Error.log, com todos os erros ocorridos durante a compilação.

3 - Algumas Vantagens do xHarbour:

a) Compatibilidade 100% com o CA-Clipper, ou seja, se seu programa não utiliza bibliotecas de terceiros, a migração é instantânea, sem ter que mexer em nenhuma linha do código.

b) Portabilidade, ou seja, o mesmo código-fonte pode ser compilado tanto para rodar em DOS, Windows (98, 2000, XP, CE, etc.), Linux (32/64 bits), Unix (32/64 bits), Mac OS X, Pocket PC, etc. (exceto se você usar funções do xHarbour que conversam com bibliotecas próprias do sistema operacional. Por exemplo: se você resolver fazer seu sistema se comunicar com a API do Windows, ao compilá-lo para Linux, esta parte do seu programa não vai funcionar, obviamente, inviabilizando assim a portabilidade).

c) Possibilidade de imprimir nas impressoras do Windows, sejam elas USB, paralelas, fax, etc.

d) Possibilidade de se trabalhar com DLLs.

e) O executável gerado será de 32 bits (e não mais 16 bits como no Clipper), podendo até ser de 64 bits, dependendo do sistema operacional.

f) Possibilidade de se manter a mesma aparência texto do Clipper ou trabalhar com telas gráficas (idênticas às do Delphi, por exemplo), valendo-se de GUIs como HwGUI, MiniGUI, etc.

g) Possibilidade de se trabalhar com bancos de dados relacionais (Firebird, MySQL, etc.), através de bibliotecas como a SQL LIB, por exemplo.

h) Possibilidade de programação orientada a objetos, exatamente como se programava no CA-Clipper com a biblioteca Class(y). No xHarbour, isto se dá através da GUI ooHG.

i) Diferentemente do Clipper, o xHarbour não “devora” o processador em ambientes Windows NT/XP/2000.

j) Ainda diferentemente do Clipper, o xHarbour reconhece arquivos com nomes longos. Aquele velho problema do Clipper de só reconhecer arquivos no formato 8.3 acaba.

k) Várias funções que não existiam ou deixaram de funcionar no Clipper quando as máquinas e os sistemas operacionais passaram a ser mais modernos agora existem de forma nativa no xHarbour. Veja alguns exemplos:

- Existem no xHarbour as funções SX_FCompress( ) e SX_FDecompress( ), ou ainda, as funções HB_Compress( ) e HB_DCompress, que comprimem e descomprimem arquivos, como os tão famosos utilitários PKZip e PKUnzip.

- A função Volume( ) do xHarbour retorna o nome do volume do HD.

- A função VolSerial( ) do xHarbour retorna o serial do HD.

- Funções nativas de mouse estão disponíveis (MShow( ), MHide( ), MRow( ), MCol( ), etc.).

- Funções de hash MD5 agora estão disponíveis através das funções HB_MD5( ) e HB_MD5File( ) do xHarbour.

4 - Instalando o xHarbour:

Primeiramente, é preciso baixar da internet os seguintes 3 arquivos (em www.xharbour.org e www.sqllib.com.br você encontra estes instaladores):

a) Link para baixar o compilador Borland C++ 5.5.1 (ele pega o arquivo .C gerador pelo xHarbour e gera o executável em C):

http://cc.codegear.com/Free.aspx?id=24778

b) Link para baixar o compilador xHarbour 1.0.0 (ele transforma um .PRG em .C, ou seja, de Clipper para C):

http://sourceforge.net/project/downloading.php?groupname=xharbour&filename=xharbour-1-0-0-beta1-bin-w32-bcc-5-5.exe&use_mirror=ufpr

c) Link para baixar a IDE xDevStudio 0.72 – Black Edition (criar, compilar, gerenciar projetos, etc.):

http://www.sqllib.com.br/v4/index.php?artigo=xDevStudio&page=Download

4.1 – Instalando o Compilador Borland C++ versão 5.5.1

Instale primeiro o compilador Borland C++ - versão 5.5.1, alterando a pasta sugerida de instalação para C:\bcc55.

Então, busque pelo arquivo harbour_cfg.zip na internet, baixe-o e salve seu conteúdo (bcc32.cfg e ilink32.cfg) dentro da pasta C:\bcc55\bin. Posteriormente, abra estes dois arquivos .CFG e garanta que os paths informados dentro deles são os mesmos onde você instalou os compiladores.

4.2 – Instalando o Compilador xHarbour

Em seguida, instale o compilador do xHarbour.org (xHarbour – versão 1.0.0 - SimpLex), alterando a pasta sugerida de instalação para C:\. Não aceite a opção "C:\xHarbour" sugerida. Infelizmente, este é um bug do instalador do xHarbour. Se você deixa a sugestão "C:\xHarbour", o instalador monta a seguinte árvore:

C:\xHarbour\xHarbour\BIN
C:\xHarbour\xHarbour\DOC
C:\xHarbour\xHarbour\INCLUDE
C:\xHarbour\xHarbour\LIB

Mas se você informa apenas C:\, a árvore fica correta, veja:

C:\xHarbour\BIN
C:\xHarbour\DOC
C:\xHarbour\INCLUDE
C:\xHarbour\LIB

No mais, a instalação da xHarbour, assim como a do Borland C++, é simples, sem mistérios.

4.3 – Instalando a IDE xDevStudio

Por último, instale a xDevStudio versão 0.72 (Black Edition). A instalação também é muito simples, sem complicações.

Porém, após instalá-la, é preciso configurá-la de forma que se possa cadastrar seus antigos projetos em Clipper e fazê-los compilar e rodar.

Toda esta configuração não é muito simples para quem está começando. Por isto, criei um tópico muito bem explicado, um passo-a-passo com imagens e tudo o mais. Ele está mais adiante.

4.4 – Configurando o Ambiente do Windows XP

Para finalizar, é preciso criar ou alterar algumas variáveis de ambiente no Windows para que o xHarbour compile e rode seus programas.

Abaixo, relacionamos as variáveis de ambiente que devem ser criadas no AUTOEXEC.NT e no CONFIG.NT (localizados em C:\Windows\System32), além de também em Painel de Controle → Sistema → Avançado → Variáveis de Ambiente.

4.4.1 - Inclua em "Variáveis de Ambiente", no Painel de Controle:

SET PATH=%PATH%;C:\xharbour\bin;C:\bcc55\bin

SET HB_PATH=C:\xharbour

SET INCLUDE=C:\xharbour\include;C:\bcc55\include

SET LIB=C:\xharbour\lib;C:\bcc55\lib
SET OBJ=C:\xharbour\lib;C:\bcc55\lib

4.4.2 - Inclua no AUTOEXEC.NT (localizado em C:\Windows\System32):

MODE CON COLS=80 LINES=25

SET PATH=%PATH%;C:\xharbour\bin;C:\bcc55\bin

SET HB_PATH=C:\xharbour

SET INCLUDE=C:\xharbour\include;C:\bcc55\include

SET LIB=C:\xharbour\lib;C:\bcc55\lib

SET OBJ=C:\xharbour\lib;C:\bcc55\lib

4.4.3 - Inclua no CONFIG.NT (localizado em C:\Windows\System32):

FILES=255
BUFFERS=99

5 – Configurando a xDevStudio 0.72 Black Edition:

Após a instalação da IDE xDevStudio, é necessário realizar 3 etapas:

a) Cadastrar compiladores (os que você já possuir instalados em sua máquina)
b) Criar os projetos (cadastrar os que já existem ou criar novos)
c) Atribuir compiladores já cadastrados na xDevStudio a cada projeto criado

A seguir, explicaremos como cadastrar compiladores na xDevStudio (ex.: xHarbour com Borland C++, xHarbour com Microsoft Visual C++, Harbour com MiniGUI,, Harbour com HwGUI, Clipper 5 com Blinker/RTLink, Harbour com FiveWin e Borland C++, etc.).

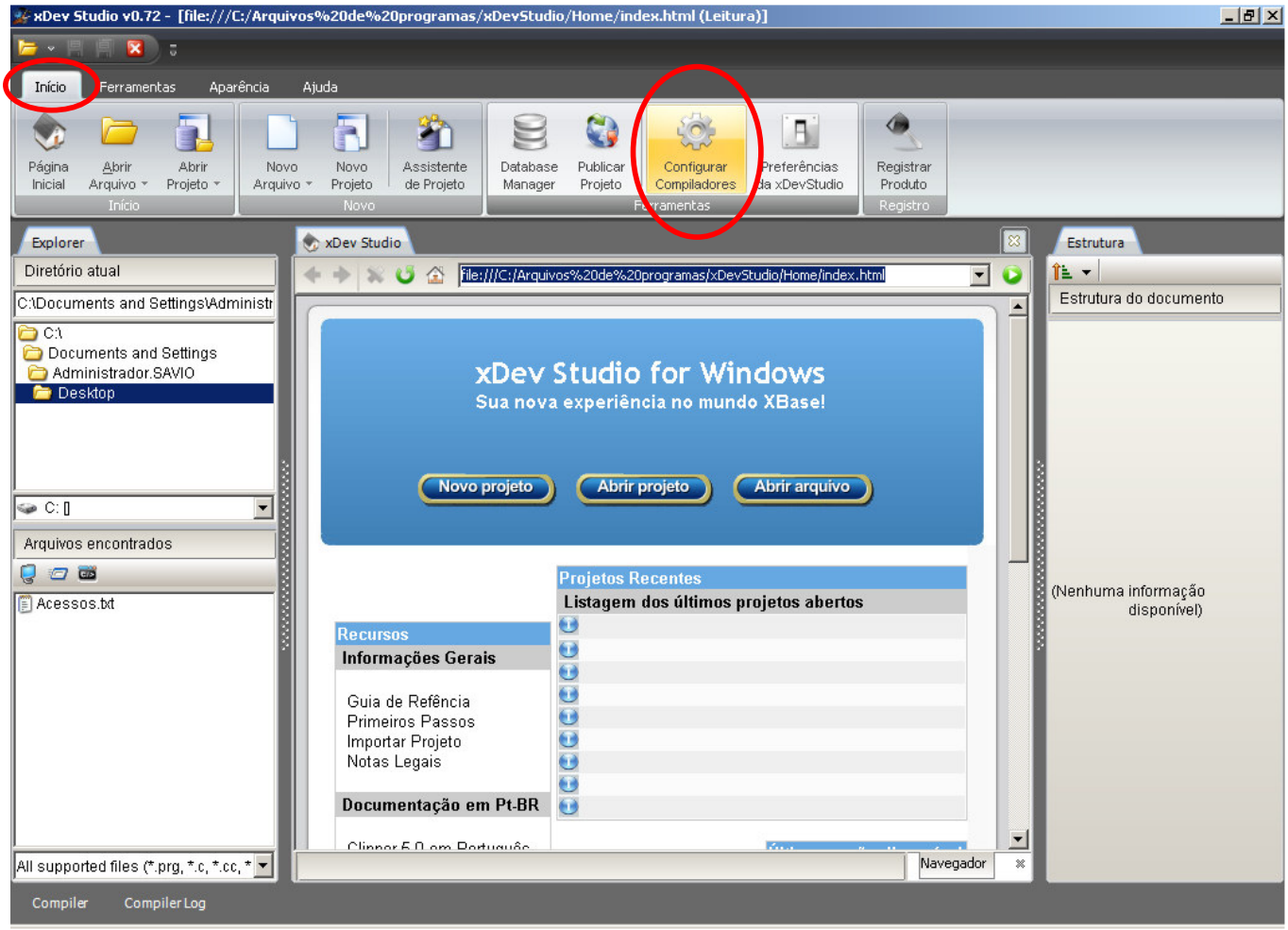
Após cadastrar quantos compiladores quiser, cada projeto poderá ser compilado com um compilador diferente. Este é o papel da xDevStudio: gerenciar projetos compilados com os mais variados compiladores.

Em seguida mostraremos como é fácil criar um projeto.

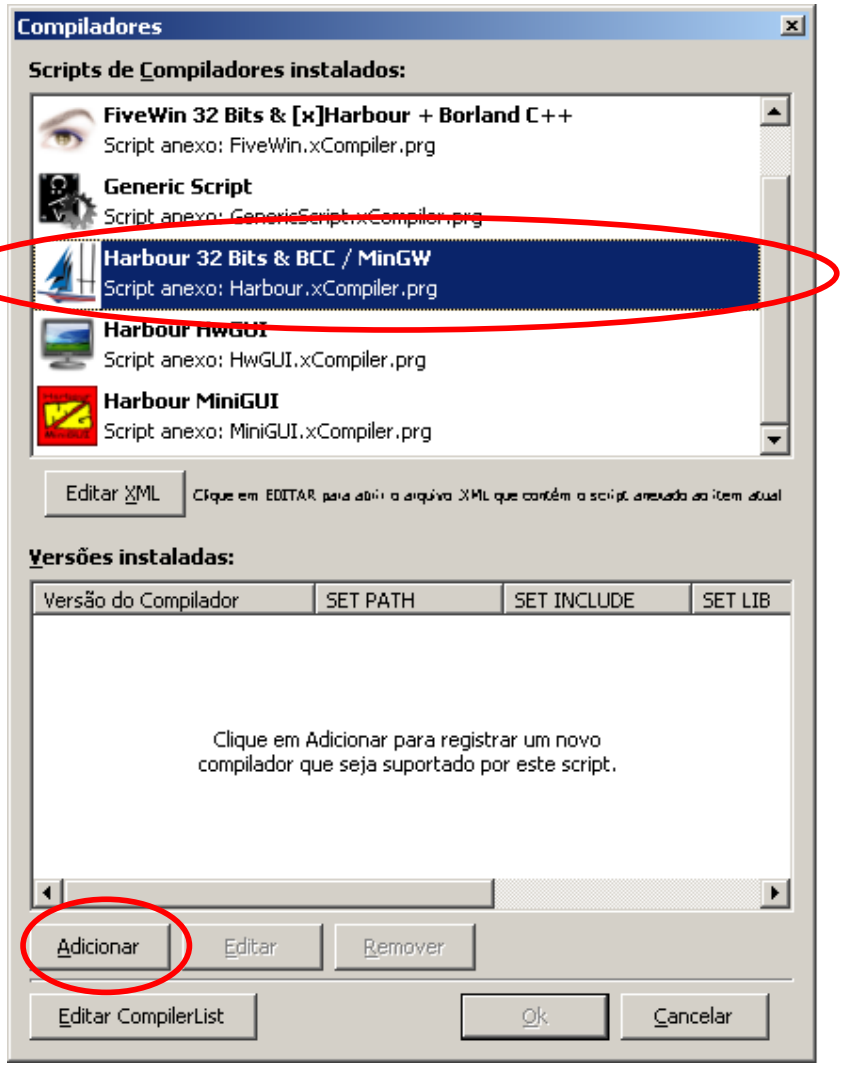
Para encerrar, mostraremos como atribuir um compilador já cadastrado na xDevStudio ao novo projeto. Veja:

5.1 - Cadastrando Compiladores na xDevStudio

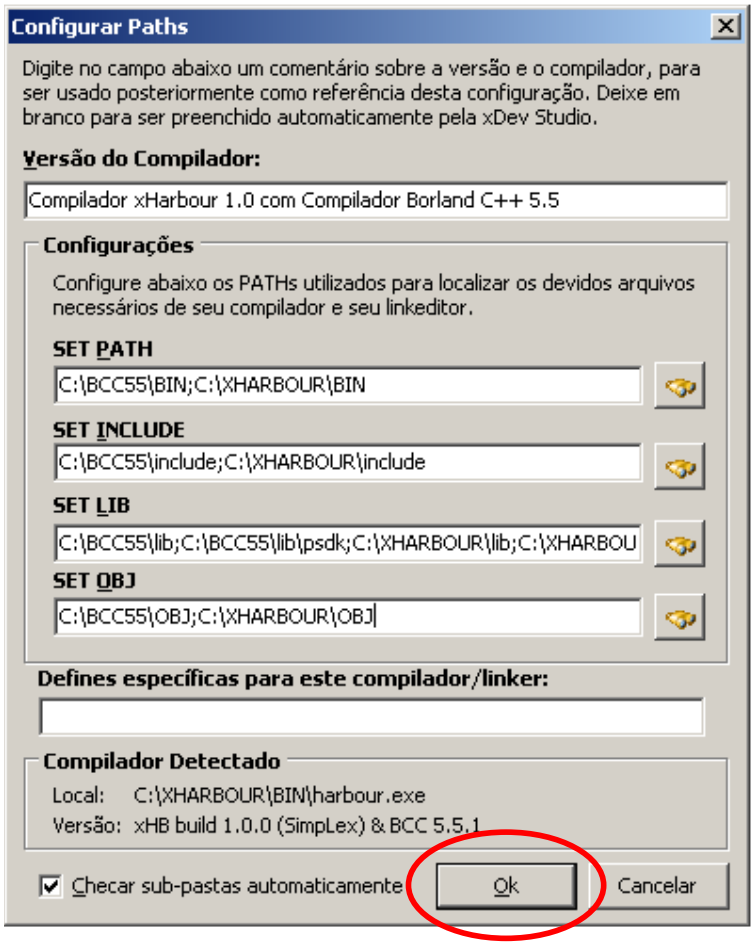
1º - Na aba “Início”, clique no botão “Configurar Compiladores”, em amarelo na foto abaixo:



2º - Em seguida, aparecerá a tela abaixo. Então, selecione a opção de script que mais se encaixa à dupla de compiladores que deseja cadastrar. Como vamos cadastrar os compiladores xHarbour 1.0 com Borland C++ 5.5, vamos selecionar a opção “Harbour 32 Bits & BCC / MinGW” e clicar no botão “Adicionar”:



3º - Ao clicar no botão “Adicionar” na tela anterior, a seguinte tela será exibida, solicitando os caminhos (paths) onde se encontram os arquivos da dupla de compiladores que você está instalando. Preencha conforme a imagem e clique em “Ok” para finalizar:

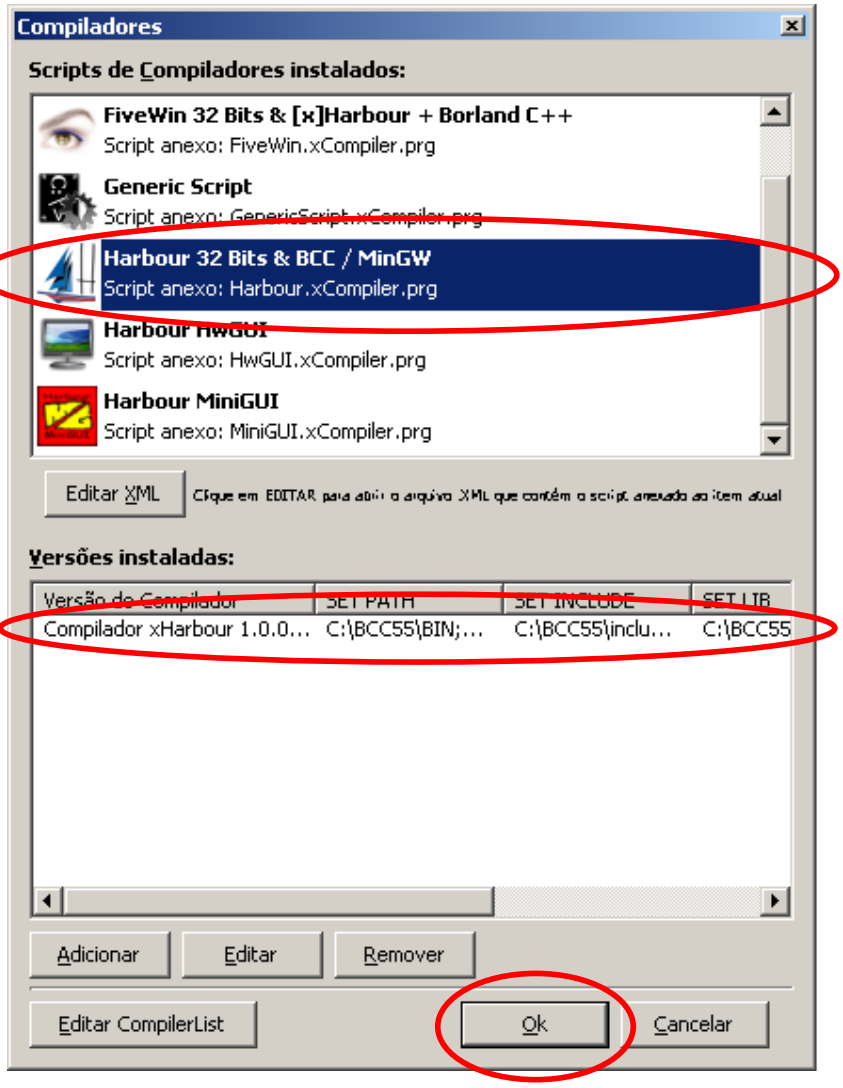


Observações:

a) Em “Versão do Compilador”, dê um nome qualquer (a seu gosto) à nova dupla de compiladores que você está instalando. Normalmente, usamos os próprios nomes da dupla de compiladores (no exemplo acima, “Compilador xHarbour 1.0 com Compilador Borland C++ 5.5”), pois isto facilita a identificação.

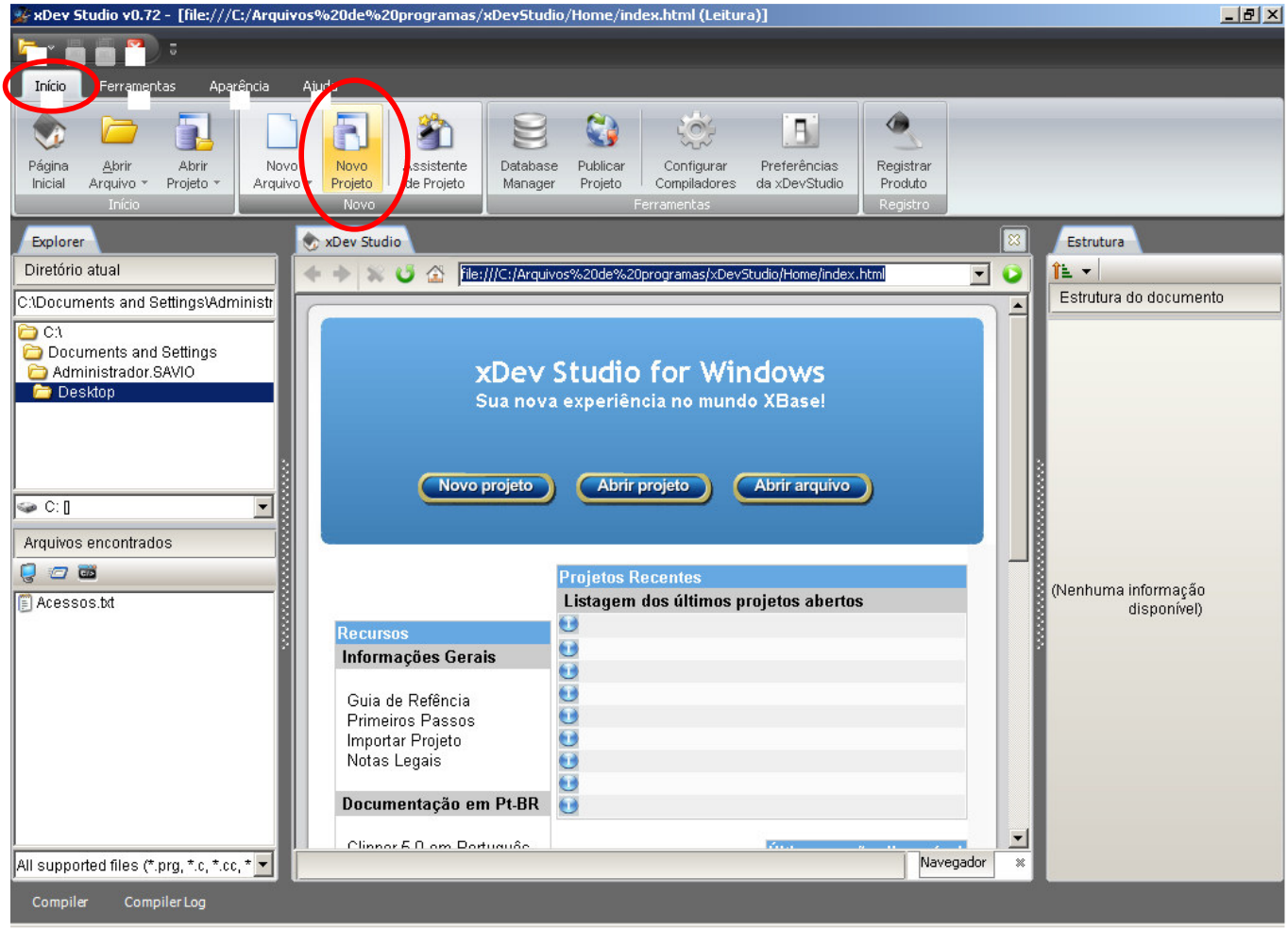
b) Em SET PATH, adicione apenas as pastas “C:\BCC55\Bin” (pasta onde se encontra o compilador “C” instalado) e “C:\xHarbour\Bin” (pasta onde se encontra o compilador xHarbour instalado), separadas sempre por ponto-e-vírgula. Caso os compiladores tenham sido instalados em pastas diferentes, aponte para lá, sem se esquecer de indicar a subpasta BIN, pois, do contrário, a xDevStudio não reconhecerá seus compiladores. Faça o mesmo com os demais paths, como exibido na imagem.

4º - Ao clicar no botão “Ok” na tela anterior, a seguinte tela voltará a ser exibida, já com a dupla de compiladores devidamente adicionada na parte de baixo da janela. Clique em “Adicionar” caso queira adicionar mais compiladores ou simplesmente clique em “Ok” para finalizar o cadastramento dos compiladores na xDevStudio:

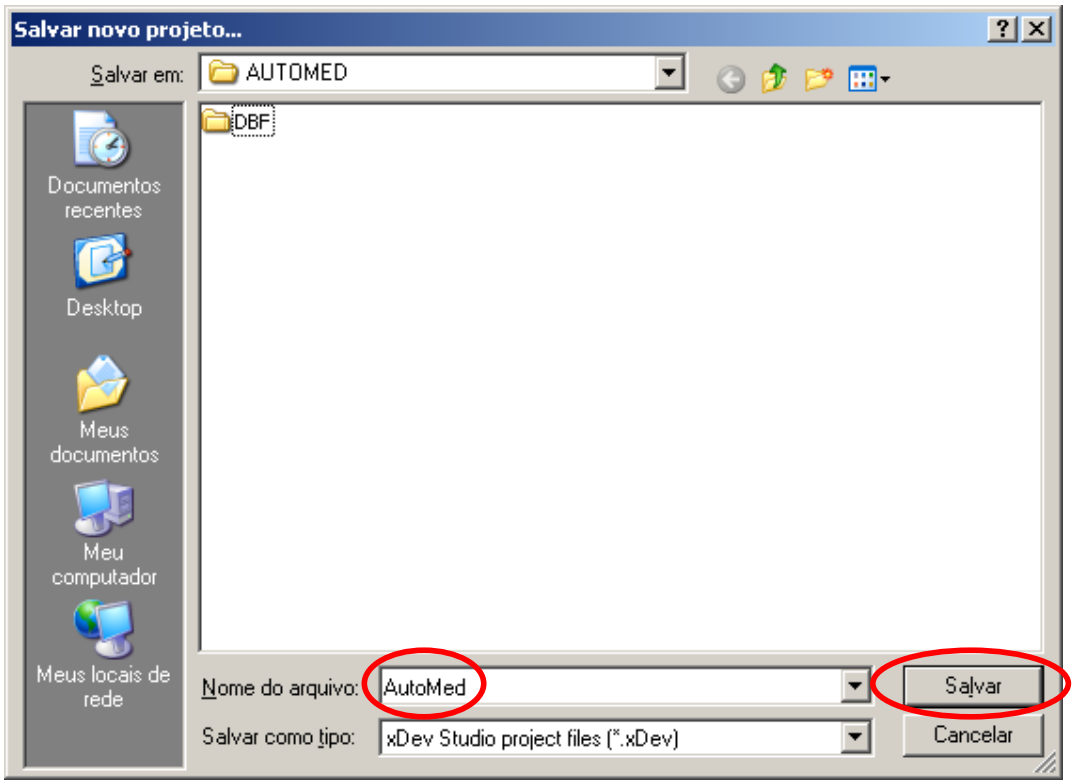


5.2 – Criando um Projeto

1º - Na aba “Início”, clique no botão “Novo Projeto”, em amarelo na foto abaixo:

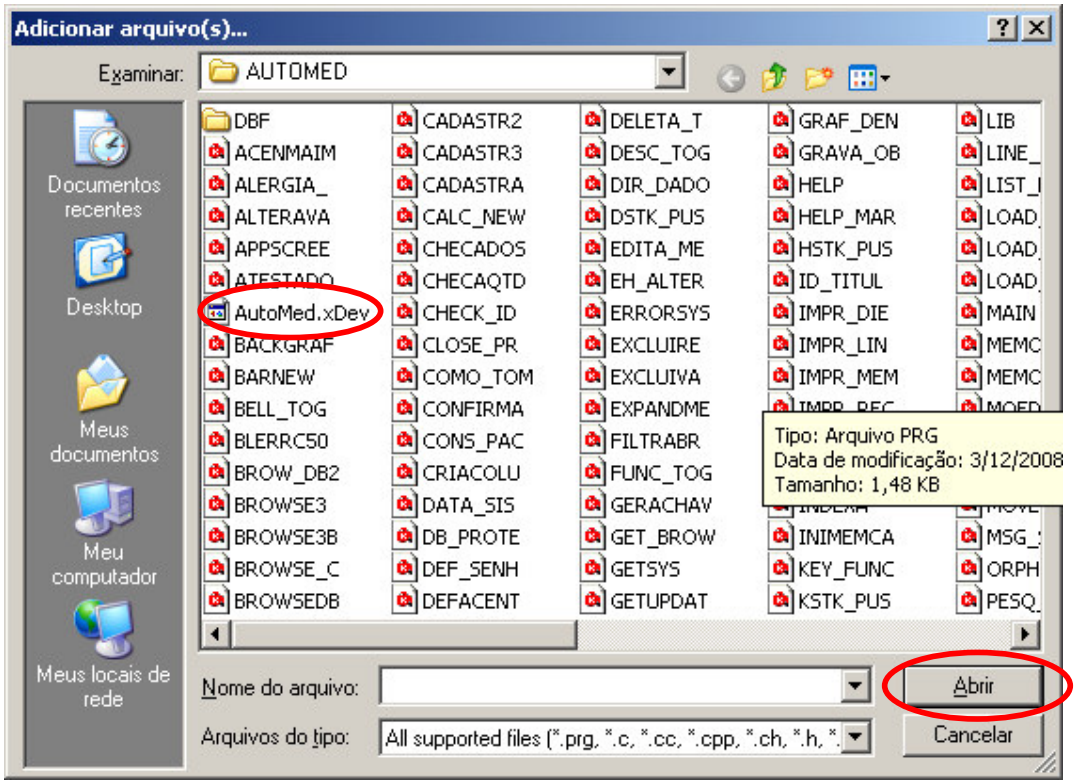


2º - A tela abaixo será exibida, solicitando que você aponte para a pasta onde se encontram os arquivos do seu projeto. Aponte para onde estão os .PRG, dê um nome qualquer a seu projeto, escrevendo-o na caixa “Nome do arquivo” e clique em “Salvar”:



3º - Em seguida, a tela abaixo será exibida para que você adicione arquivos ao seu projeto. Não é necessário adicionar somente .PRGs. Você pode (e deve) adicionar os .CH também, além de .TXT, .DBF, etc. Eu costumo adicionar apenas os .PRGs e os .CH.

Selecione quantos e quais arquivos desejar e clique em "Abrir" para adicioná-los ao projeto.

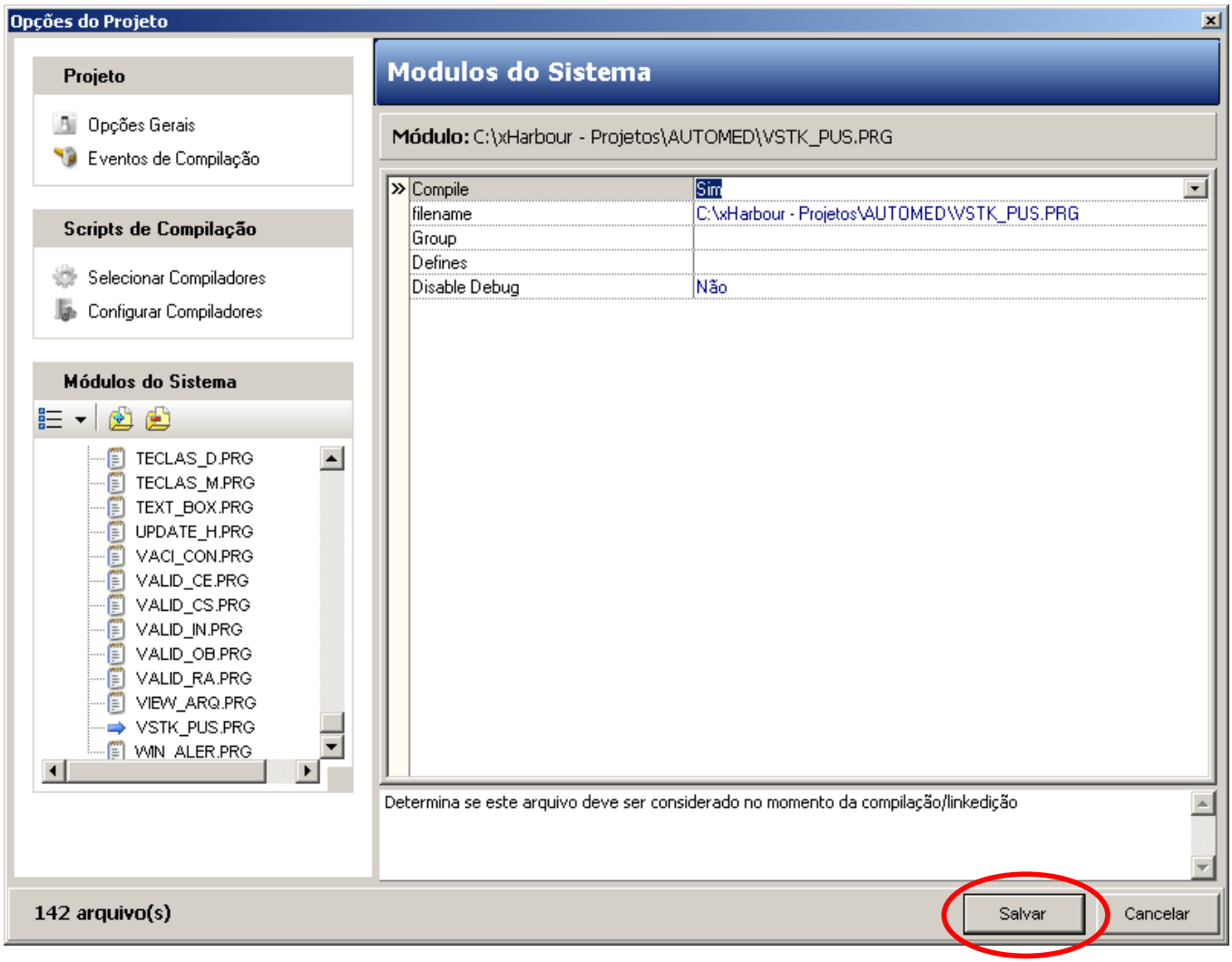


Observações:

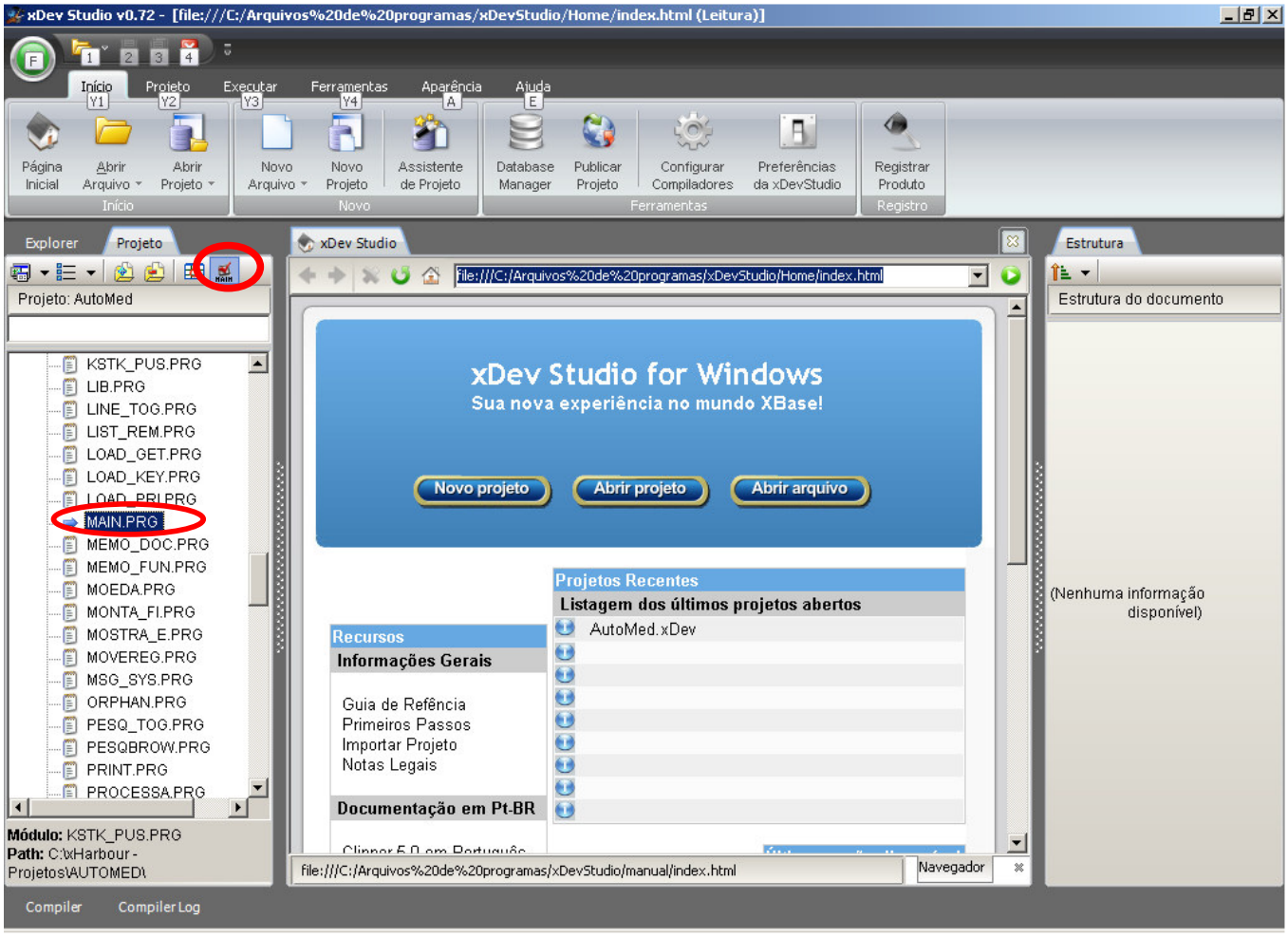
a) Repare no detalhe que o projeto foi criado com o nome dado na tela anterior e com a extensão .xDev. Este é o arquivo que contém as configurações de seu projeto. Vale a pena abri-lo e conhecê-lo.

4º - A seguinte tela será apresentada. Trata-se da configuração individual de cada .PRG adicionado ao seu projeto. Repare que no exemplo abaixo, ela se refere ao módulo "C:\xHarbour – Projetos\AUTOMED**VSTK_PUS.PRG**". Você pode informar se este módulo deverá ser compilado ou não, se deverá ser debugado ou não, etc.

Normalmente, não mudamos nada nesta tela. Simplesmente clicamos em "Salvar" e pronto. Seu projeto está criado na xDevStudio:



5º - Após a criação do projeto, falta apenas indicar qual dos PRGs é o que contém a função Main( ), obrigatória no xHarbour. Para indicar, faça o seguinte: na lista de PRGs exibida na janela da esquerda, selecione o arquivo e clique no último botãozinho da direita acima da lista, chamado “Main”. Pronto. Está terminada a criação do projeto na xDevStudio:



Observações:

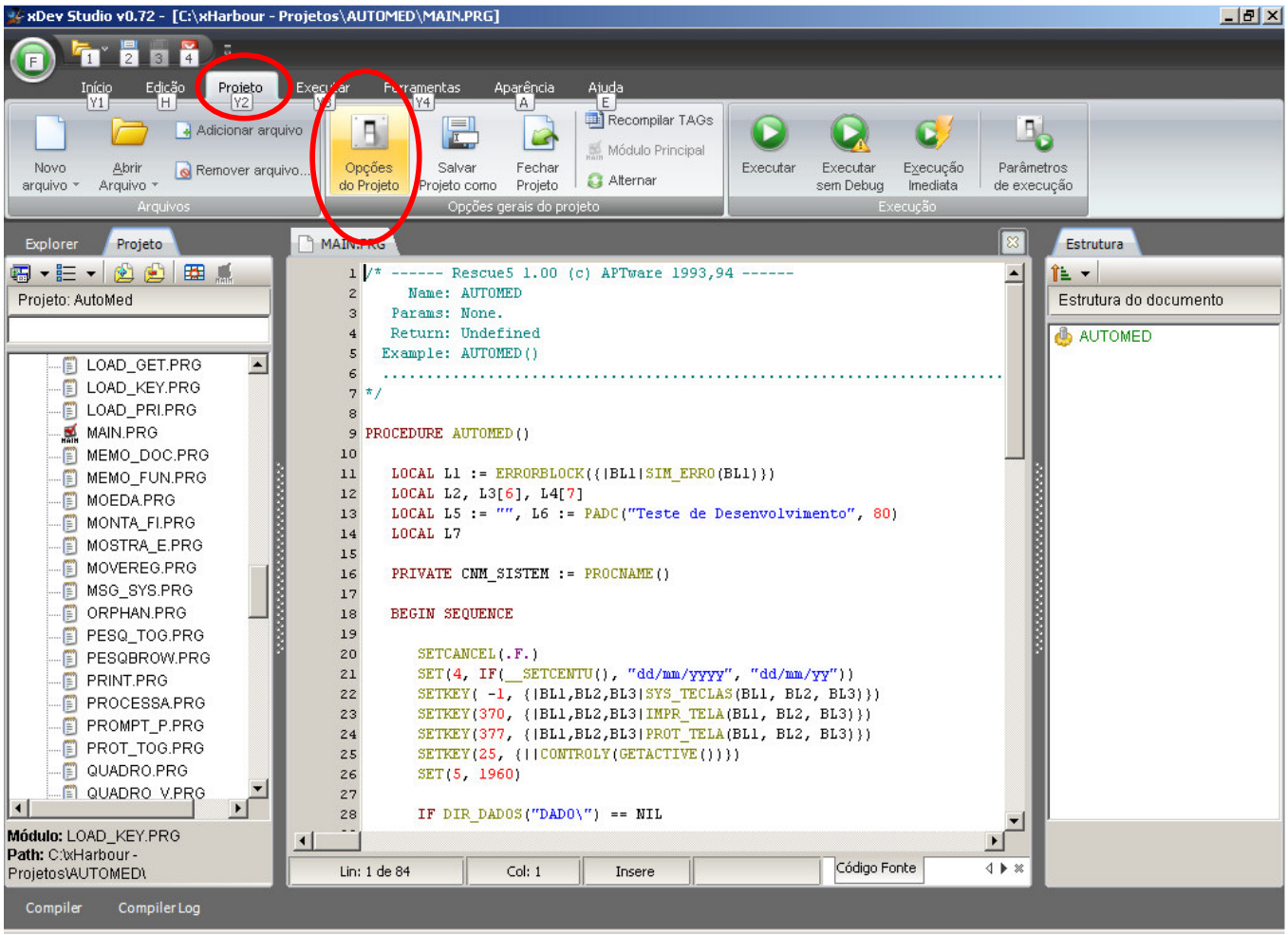
a) Para editar um arquivo, clique duas vezes sobre ele na lista da esquerda. Cada arquivo aberto será exibido em abas no centro da IDE.

b) Para expandir a área de edição, utilize o F4. Pressione F4 novamente para voltar à IDE.

c) Para compilar e executar o projeto de dentro da xDevStudio, pressione F9. Mas lembre-se: a compilação e execução com o F9 só será possível após a atribuição de um compilador ao projeto, o que será explicado a seguir.

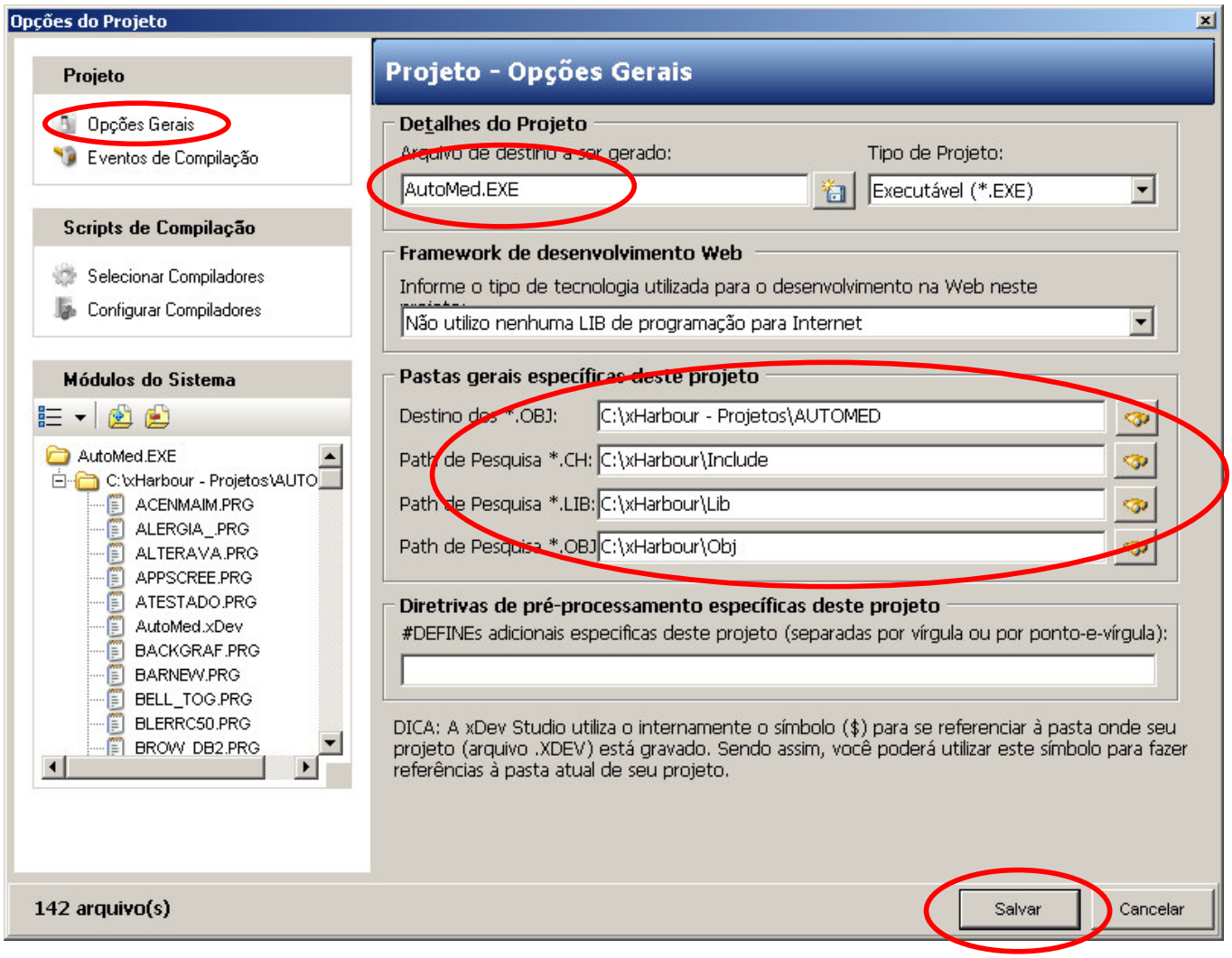
5.3 – Atribuindo um Compilador ao Projeto

1º - Com o projeto aberto, clique na aba “Projeto” e, em seguida, no botão “Opções do Projeto”, em amarelo na figura abaixo:



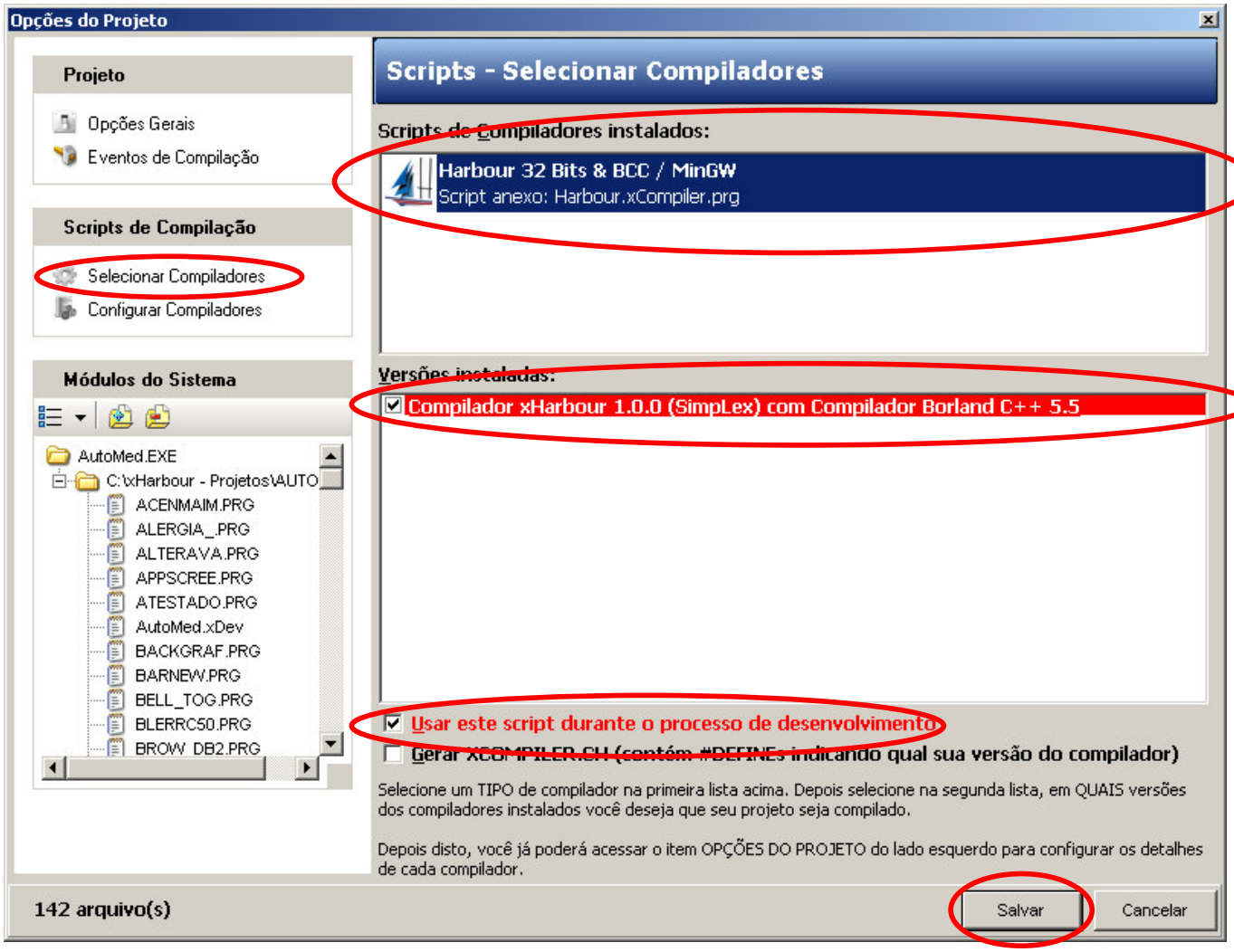
2º - Em seguida, a seguinte tela será apresentada. Repare no título em azul (“Projeto – Opções Gerais”) que esta tela se trata do primeiro tópico de configuração dos vários existentes na janelinha da esquerda. Teremos que configurar “Opções Gerais”, em seguida “Selecionar Compiladores” e, por último, “Configurar Compiladores” (mas isto veremos mais adiante).

Bem, vamos concluir a fase “Opções Gerais” agora. Indique o nome do executável a ser gerado, preencha o destino dos OBJs a serem gerados para este projeto (pode ser em qualquer pasta a seu gosto, mas normalmente usamos a própria pasta do projeto) e os paths onde se encontram os .CH, .LIB e .OBJ que este projeto utiliza. Para finalizar, clique em “Salvar”:



3º - Agora repita o 1º passo e clique em “Selecionar Compiladores”, na janelinha da esquerda mostrada na figura anterior.

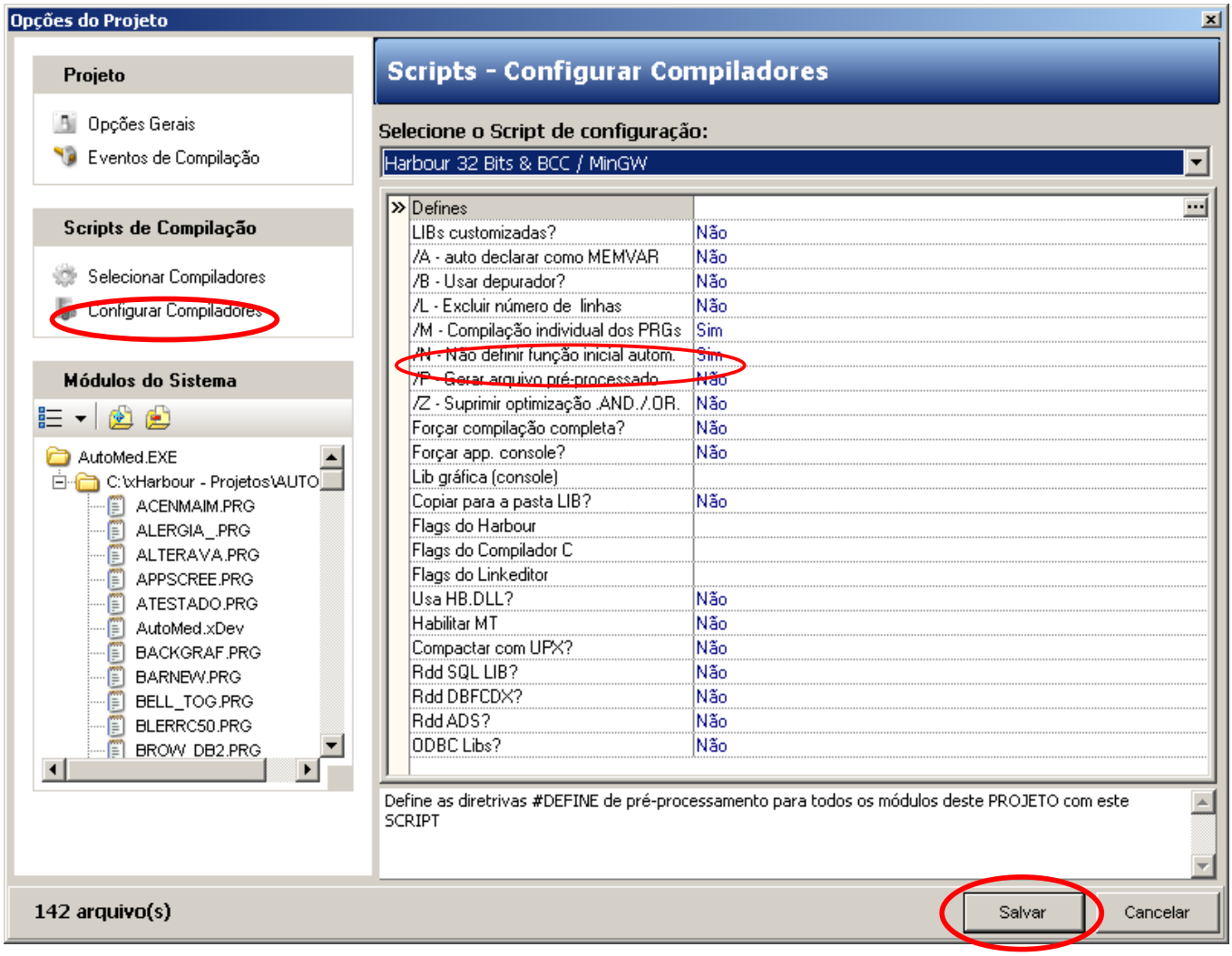
A seguinte tela aparecerá. Selecione o compilador desejado, marque-o na janela de baixo (“versões instaladas”) e não se esqueça de marcar também a opção “Usar este script durante o processo de desenvolvimento”. Clique em “Salvar”:



4º - Novamente, repita o 1º passo e desta vez clique em “Configurar Compiladores”, na janelinha da esquerda mostrada na figura do 2º passo.

A seguinte tela aparecerá. Garanta que a opção “/N” (Não definir função inicial automaticamente) esteja com “Sim”. Só ficaria com “Não” se seu projeto não tiver a função Main( ), mas isto não é recomendável. Clique em “Salvar”.

Pronto. O compilador está atribuído ao seu projeto. Basta usar F9 para compilar e executar de dentro da xDevStudio.



6 – Dicas & Observações:

6.1 – Sobre a xDevStudio

a) Para editar um arquivo, clique duas vezes sobre ele na lista da esquerda. Cada arquivo aberto será exibido em abas no centro da IDE.

b) Para expandir a área de edição, utilize o F4. Pressione F4 novamente para voltar aos controles da IDE.

c) O F9 compila e executa seu projeto de dentro da própria xDevStudio.

d) Após a criação de um projeto, a xDevStudio cria automaticamente um script de compilação para que você possa compilar seu sistema diretamente do DOS. Ele é uma BATCH criada com o nome de "Build_MyPRG.bat". Isto permite a compilação por fora da xDevStudio.

e) A xDevStudio cria também o arquivo "Error.log", que contém os erros de compilação do sistema, além de desenhar a tela no momento em que o sistema é abortado. A cada vez que você tentar compilar, este arquivo é gerado caso haja erros. Acostume-se a verificá-lo sempre.

f) Quando eu pressionava F9 na xDevStudio para compilar e executar um projeto, a execução se dava em janela minimizada do Windows. Para poder rodar em tela cheia na xDevStudio, deve-se colocar no início da função Main( ) a instrução SetMode(25, 80).

g) Uma boa dica é clicar na aba "Ajuda", depois em "Início", pressionar F4 e clicar em "Recursos → Informações Gerais → Primeiros Passos". Um documento será exibido com ainda mais informações a respeito das configurações da xDevStudio.

6.2 – Sobre as Primeiras Migrações de Clipper para xHarbour

Abaixo relato algumas pequenas "pedreiras" que enfrentei nas primeiras migrações. Vejam no que esbarrei. Resolvi tudo, mas levei um tempinho. Por isso resolvi documentar:

6.2.1 - Incompatibilidade da função DbEdit( ) entre Clipper e xHarbour:

a) O terceiro parâmetro (linha final do grid) não confere entre as duas versões. Mas o errado era o do Clipper. O xHarbour apenas o corrigiu, colocando-o dentro da área certinha do box. O problema foi que deu trabalho em trocar o parâmetro em todas as chamadas de DbEdit( ) nos meus códigos-fontes.

b) Outro problema (e este, na verdade, eu não resolvi, apenas o contornei) foi que o DbEdit( ) do xHarbour não apresenta o rodapé de forma correta. Mais precisamente a linha separadora do rodapé. Briguei com ele por quase 1 hora e então resolvi dar um "jeitinho brasileiro", diminuindo a área de abrangência do grid e escrevendo a linha separadora "na mão", fazendo assim o rodapé. Isto deu trabalho e ficou sem a solução limpa que eu esperava encontrar.

6.2.2 – Alteração no tamanho dos nomes de variáveis:

Sabemos que no Clipper uma variável pode ter seu nome com quantos caracteres quiser, mas somente os 10 primeiros são considerados pelo compilador. No xHarbour é diferente. Eu não sei quantos caracteres o xHarbour reconhece, mas apanhei muito com um erro meu mesmo num código-fonte, mas que passou despercebido no Clipper.

Eu tinha uma variável de nome "nValTomOriginal" declarada como "Local" dentro de uma função e, em seguida, eu a referenciava como "nValTomOrig". Isto não dava problemas no Clipper, pois

mesmo “nValTomOrig” já tinha mais de 10 caracteres. O compilador Clipper nunca reclamou, é claro. Quando compilei com o xHarbour, o compilador também não reclamou de nada, mas na execução apresentou “Variable does not exist”. Bem, demorei pra achar o problema, mas ao identificá-lo, foi fácil corrigí-lo.

6.2.3 – Utilização de Picture nos @Say..Get:

Sabemos que uma Picture tem que ser expressa no Get sempre entre aspas, mas eu tinha um único @Say..Get no meu sistema com variável numérica, cuja Picture estava expressa como 99.99 e não “99.99”. O fato de estar sem as aspas retornou um erro super esquisito no xHarbour, ininteligível, que em nada ajudou a descobrir o problema. Eu olhava e reolhava, sem perceber nada de errado com o código. Só depois de um tempo, ao conferir o Get com outro de outra função, foi que percebi que havia me esquecido das aspas. O erro que o xHarbour apresentou foi:

“Argument Error: AT Arguments ([1] = Type: C Val: @ [2] = Type: N Val: 99.99)”.

O Clipper aceitava essa declaração sem as aspas, funcionando corretamente. O xHarbour não aceita. Vale a pena ficar ligado, pois as mensagens de erro apresentadas pelo xHarbour não são tão esclarecedoras como as do Clipper. Aliás, isso é um problema do xHarbour, na minha opinião. As mensagens de erro podiam ser melhores, mais esclarecedoras.

6.2.4 – Comparação envolvendo a função IndexExt( ):

A função IndexExt( ) do xHarbour retorna a extensão ora em maiúsculas, ora em minúsculas (ex.: ".NTX" ou ".ntx"), dependendo de como o arquivo foi criado. Isto por causa do Linux, que é case sensitive. No Clipper, retorna sempre em maiúsculas. Isto pode parecer bobagem, mas eu tinha uma situação em um dos meus sistemas onde eu comparava a string “.NTX” com IndexExt( ), que retornava “.ntx” e causava erro na lógica. Corrigi o problema comparando a string “.NTX” com Upper(IndexExt( )).

6.3 – Dicas e Observações Diversas

a) Evite usar funções do xHarbour que se comunicam com a API do Windows ou features específicas de um sistema operacional para não inviabilizar a portabilidade, que é uma das grandes vantagens do xHarbour.

b) Ainda com o propósito de não inviabilizar a portabilidade de seu código-fonte, nunca coloque os paths dos arquivos acessados pelo sistema de forma hard coded, ou seja, “cravados” no código-fonte. Crie uma lista de constantes num arquivo .CH para os caminhos no padrão do Windows e outra lista com as mesmas constantes com os paths com padrão Linux. Então, antes de compilar, comente um bloco e descomente o outro, dependendo do sistema operacional que você pretende utilizar, ou simplesmente faça uso de “ifdef” para comutar entre as duas opções.

c) Para indexar nomes com acentos gráficos da língua portuguesa, insira as seguintes linhas no início do seu programa:

```
REQUEST HB_CODEPAGE_PT850
HB_SETCODEPAGE("PT850")
```

d) Se seu antigo sistema em Clipper não utiliza mouse, acrescente dentro da função Main( ) a chamada da função MHide( ) ou MSetCursor(.F.) do xHarbour, que irá desligar o mouse. Se ainda assim o ponteiro do mouse insistir em aparecer, acrescente a seguinte instrução no início do seu PRG principal:

```
REQUEST HB_NOMOUSE
```

e) Visite sempre www.xharbour.org e www.sqllib.com.br para baixar novas versões do compilador xHarbour e da xDevStudio, além de conhecer novas bibliotecas de terceiros e ficar por dentro das novidades do mundo xBase.

f) Se seu programa abre muitos arquivos, experimente adicionar ao seu projeto a biblioteca bcc640.lib. Ela permite que sejam abertos mais de 640 arquivos por sua aplicação. Busque por ela na internet e coloque-a na pasta C:\xHarbour\lib.

7 - Experiência Pessoal:

Ano passado, 2007, a empresa onde trabalho adquiriu o Flagship (o pior é que fui eu que escolhi esta ferramenta) e tentamos converter alguns sistemas utilitários em Clipper para rodar nas cerca de 1.000 estações Linux Debian, através dele. Foi um desastre. Até hoje não sei se o Flagship não é bom ou se o Linux Debian é que é complicado. Só sei que os dois juntos não se entenderam. Daí em diante me desencantei com qualquer ferramenta que propusesse migrar o Clipper para 32 bits.

Então, no início de setembro de 2008, um colega de trabalho, que é programador Delphi (ele nunca programou em Clipper) pesquisou em casa sobre o xHarbour, baixou sua versão free (**xHarbour.org**) e mais algumas bibliotecas e me mostrou tudo lá no trabalho mesmo, em seu lap top.

Eu, na hora, torci o nariz. Mas ele me convenceu a testar e resolvemos instalar o aparato na minha máquina do trabalho mesmo, já que o programador Clipper era eu.

Apanhamos para fazer a IDE xDevStudio funcionar. A xDevStudio é como se fosse a IDE do Delphi. Deveria haver um passo-a-passo explicativo de instalação junto com ela, mas não há. Aliás, em tudo o que se refere ao **xHarbour.org** (versão free do xHarbour), documentação é praticamente inexistente. Penamos bastante com aquelas coisas de configurar compilador, etc. Mas tudo deu certo ao final e fizemos o famoso "Hello World!", compilado em xHarbour e rodando de dentro da xDevStudio.

Bem, considerando que tive meu primeiro contato com o xHarbour na última segunda-feira (08/09/2008) e em menos de 1 semana já migrei 3 sistemas antigos em Clipper, não dá para torcer o nariz. E olha que eles usavam várias bibliotecas de terceiros. Estou empolgado com o xHarbour. Ganhei uma sobrevida em sistemas antigos que já não tinha esperanças disto acontecer.

Estou usando a xDevStudio 0.72 – Black Edition (que tem alguns bugs que até já reportei ao seu "dono", por ser uma versão Beta – não sei se ele leu o e-mail que lhe enviei), o xHarbour 1.0.0 (SimpLex) e o Borland C++ 5.5.1. A instalação de tudo isto foi super simples. Setadas as variáveis de ambiente no Win/XP, o resto fluiu numa boa.

Em seguida, percebi que o arquivo NG (Norton Guide) do xHarbour é super pobre e não possui todas as suas funções. A documentação do xHarbour é fraca e carecemos sempre da ajuda de um ou outro colega. Então, "caçando" na internet (e não foi muito fácil), consegui o arquivo "xHarbour Language Reference Guide.chm", que, na verdade, não é o guia de referência do **xHarbour.org** (free), e sim do **xHarbour Builder** (versão comercial, que é paga), mas já ajuda. Aí sim eu pude evoluir. Daí em diante tudo ficou bem mais fácil.

Bem, além de corrigir funções do Clipper que já não funcionavam, como as funções de espaço em disco total, livre e usado, como as funções de leitura de memória, como a impressão em USB e o consumo excessivo do processador em máquinas com Win/NT ou XP, etc., o xHarbour abriu centenas de outras possibilidades.

Muito mais contente ainda eu fiquei em ver que consegui substituir praticamente todas as minhas bibliotecas de terceiros por funções nativas do xHarbour. E tudo isto em apenas 6 dias.