# SmartHeap™ for SMP

## 入門/プラットフォーム ガイド

**対応 OS:**

**Windows Server 2003**
**Windows XP**
**Windows 2000**
**Windows NT Intel**

Version 7

Printed March 15, 2006

このマニュアルに関するご意見を下記までお寄せ下さい。

MicroQuill Software Publishing, Inc.
815 6th Street South, Suite 111
Kirkland, Washington 98033

Voice: (425) 827-7200
Fax: (425) 828-4088
Internet: info@microquill.com

# 目次

# SmartHeap 7 の新機能

SmartHeap 7 では、Visual Studio .NET の SmartHeap サポートがさらに向上しています。アンマネージの C/C++ プロジェクトに対し、静的リンクおよび動的リンクの両方を行うことができます。Debug SmartHeap により、エラー レポートのファイルと行番号を VC7 pdb データから検索できるようになりました。また、インテル社のハイパースレッディング テクノロジがサポートされています。

さらに、Itanium のハードウェアで動作する 64 ビットの Windows に対応したライブラリが含まれています。

# SmartHeap 6 の新機能

## 強化されたデザイン

SmartHeap 6.0 では、絶対速度が向上し、またメモリをより効率的に利用できるようになりました。SmartHeap 5 とは異なり、メモリは OS に返されます。その際、ユーザーは返すメモリの量と返すタイミングを管理できます。

SmartHeap 6.0 で採用されている新しい API は、主として UNIX の **`valloc`** および **`memalign`** に対応しています。**`MemAllocAligned (MEM_POOL pool、unsigned long size、unsigned long alignment、unsigned long flags)`** は、結果が「Alignment」パラメータで指定された値の倍数にアラインされる点を除き、基本的に **`MemAllocPtr`** と同じです。

## 新しいチューニング API

このリリースで導入された新しいチューニング API

- **`void MemProcessSetFreeBytes (unsigned long bytes):`** Smartheap で保持する大きなヒープ ブロック内の空き領域の容量をコントロールします。SmartHeap 5 では空き領域に

制限はなく、メモリが OS に返されることはありませんでした。SmartHeap 6 を使用することにより、この API で OS にメモリを返すタイミングを正確にコントロールすることができます。デフォルト値は 10MB です。これは大きなヒープ ブロックに 10MB 以上の空き領域がないと、メモリが OS に返されないことを意味します。値を大きく設定すると、アロケーション機能は向上しますが、フットプリントが大きくなる可能性があります。

- **`void MemProcessSetLargeBlockThreshold (unsigned long bytes):`** SmartHeap で保持する大きなヒープ ブロック内のブロック サイズをコントロールします。デフォルト サイズは 512KB です。しきい値より大きいブロックは OS に割り当てられ、直接 OS に返されます。値を大きく設定すると、性能は向上しますが、フットプリントが大きくなる可能性があります。

- **`unsigned long MemPoolSetFreeBytes (MEM_POOL pool、unsigned long bytes): MemProcessSetFreeBytes`** が大きなヒープ ブロックで空き領域をコントロールするのと同様に、プール内の空き領域をコントロールします。デフォルト値は 1MB です。値を大きく設定すると 16K サイズ以下のアロケーションの領域が増えるため、性能が向上し、競合が減少します。この値が 0 でない場合、SmartHeap ではプール内で指定されたサイズの合計を上限とし、空ページを残します。プールにページを追加する場合は、同じプールから解放されたページを再利用することで、グローバル ロックや大きなヒープ ブロックを割り当てることなく、ページを追加できます。

# SmartHeap をインストールする

SmartHeap インストール ディスクのファイルは圧縮されていないため、コピー プロテクトされていません。

- **ハード ディスクに SmartHeap をインストールするには:**

1. **`include`** ディレクトリのファイルをコンパイラの include ファイル パスのディレクトリにコピーします。
2. コンパイラ (**`msvc`**) に対応するライブラリ ディレクトリ内のファイルをリンカのライブラリ ファイル パスのディレクトリにコピーします。
3. **`bin`** ディレクトリのファイルを Windows NT、Windows 2000、Windows XP の **`system32`** ディレクトリ、Windows 9x **`system`** ディレクトリ、あるいは **`PATH`** 上の他のディレクトリへコピーします。
4. **（オプション）** **`source`** および **`samples`** からファイルをコピーして、SmartHeap のソースファイルおよびサンプル アプリケーションの検証や修正を行うことができます。

# Win32/Win64 SmartHeap リリースのファイル

SmartHeap 7.0 に含まれる内容は次のとおりです:

- Microsoft Visual C++ 用インポート ライブラリ。
- Microsoft Visual C++ 用静的リンク可能ライブラリ。
- コンパイラに依存しない DLL。
- プラットフォームに依存しないヘッダ ファイル。
- **`malloc`** および C++ **`operator new`** 定義に必要なソース ファイル (プラットフォーム非依存)。

SmartHeap CD に含まれる Windows 固有のファイルは次のとおりです:

| ディレクトリ | ファイル | 説明 |
|---|---|---|
| **`include`** | **`smrtheap.h`** | **`malloc`** を除くすべての Runtime SmartHeap C 宣言を含むヘッダ ファイル |
| **`include`** | **`smrtheap.hpp`** | **`operator new`** などの Runtime SmartHeap C++ 宣言を含む ヘッダ ファイル |
| **`include`** | **`shmalloc.h`** | Runtime SmartHeap ANSI C 関数 (**`malloc`** など) を含むヘッダ ファイル |
| **`include`** | **`heapagnt.h`** | すべての Debug SmartHeap 宣言を含むヘッダ ファイル |
| **`bin`** | **`shsmp.dll`** | 32 ビット版ランタイム SmartHeap ダイナミック リンク ライブラリ |
| **`bin`** | **`shsmp.dbg`** | Runtime SmartHeap DLL 用 32 ビット記号 |
| **`bin`** | **`shsmpd.dll`** | 32 ビット版デバッグ SMP 用 SmartHeap ダイナミック ライブラリ |
| **`bin`** | **`shsmpd.dll`** | Debug SMP SmartHeap ダイナミック ライブラリ用 32 ビット記号 |
| **`bin`** | **`shw32d.dll`** | 32 ビット版 デバッグ SmartHeap ダイナミック リンク ライブラリ |
| **`bin`** | **`shsmp64.dll`** | 64 ビット版ランタイム SmartHeap ダイナミック リンク ライブラリ |

| ディレクトリ | ファイル | 説明 |
|---|---|---|
| **bin** | **shsmpd64.dll** | 64 ビット版 デバッグ SMP SmartHeap ダイナミックリンク ライブラリ |
| **bin** | **shw64d.dll** | 64 ビット版デバッグ SmartHeap ダイナミック リンク ライブラリ |
| **msvc** | **shsmp.c** | 非 MFC アプリケーション (32 ビットビルドのみ) で SmartHeap を使う場合の「クイック スタート」プロシージャ用ソース ファイル |
| **msvc** | **shmfcsmp.cpp** | MFC アプリケーション (32 ビット ビルド のみ) で SmartHeap を使う場合の「クイック スタート」プロシージャ用ソース ファイル |
| **msvc** | **shlsmpmt.lib** | 静的リンク可能な 32 ビット版ランタイム SmartHeap マルチスレッド ライブラリ (Microsoft Visual C++ 用) |
| **msvc** | **shlsmpmtd.lib** | 静的リンク可能な 32 ビット版ランタイム SmartHeap マルチスレッド デバッグ SMP ライブラリ (Microsoft Visual C++ 用) |
| **msvc** | **shdsmpmt.lib** | 32 ビット版ランタイム SmartHeap マルチスレッド DLL インポート ライブラリ (Microsoft Visual C++ 用) |
| **msvc** | **shdsmpmtd.lib** | 32 ビット版ランタイム SmartHeap マルチスレッド DLL インポート デバッグ SMP ライブラリ (Microsoft Visual C++ 用) |
| **msvc\64** | **shlsmp64mt.lib** | 静的リンク可能な 64 ビット版ランタイム SmartHeap マルチスレッド ライブラリ (Microsoft Visual C++ 用) |
| **msvc\64** | **shlsmp64mtd.lib** | 静的リンク可能な 64 ビット版ランタイム SmartHeap マルチスレッド デバッグ SMP ライブラリ (Microsoft Visual C++ 用) |
| **msvc\64** | **shdsmp64mt.lib** | 64 ビット版ランタイム SmartHeap マルチスレッド DLL インポート ライブラリ (Microsoft Visual C++ 用) |
| **msvc\64** | **shdsmp64mtd.lib** | 64 ビット版 ランタイム SmartHeap マルチスレッド DLL インポート デバッグ SMP ライブラリ (Microsoft Visual C++ 用) |

| ディレクトリ | ファイル | 説明 |
|---|---|---|
| **msvc\vc6** | **shmfc4m.lib** | 静的リンク可能な 32 ビット版マルチスレッド ライブラリ ファイル (Microsoft Visual C++/MFC 4.x/5.x 用。静的リンクされた Release MFC のみ。SmartHeap ライブラリの**前に**リンクしなければなりません。) |
| **msvc\vc7** | **shmfc4m.lib** | 静的リンク可能な 32 ビット版マルチスレッド ライブラリ (Microsoft VC++ 7/MFC 4.x/5.x 用。静的リンクされた Release MFC のみ。SmartHeap ライブラリの**前に**リンクしなければなりません。) |
| **msvc** | **shdw32md.lib** | 32 ビット版デバッグ SmartHeap DLL インポート ライブラリ (Microsoft Visual C++ 用) |
| **msvc\64** | **shdw64md.lib** | 64 ビット版デバッグ SmartHeap DLL インポート ライブラリ (Microsoft Visual C++ 用) |
| **msvc\vc6** | **shmfc32md.lib** | 静的リンク可能な 32 ビット版ライブラリ ファイル (Microsoft Visual C++ 2.x/MFC 3.0 用。静的リンクされた **_DEBUG** MFC のみ。SmartHeap ライブラリの**前に**リンクしなければなりません。) |
| **msvc\vc7** | **shmfc32md.lib** | 静的リンク可能な 32 ビット版ライブラリ ファイル (Microsoft VC++ 7/MFC 3.0 用。静的リンクされた **_DEBUG** MFC のみ。Debug SmartHeap ライブラリの**前に**リンクしなければなりません。) |
| **msvc\vc6** | **shmfc4md.lib** | 静的リンク可能な 32 ビット版ライブラリ ファイル (Microsoft Visual C++ 4.x/5.x 用。静的リンクされた **_DEBUG** MFC のみ。Debug SmartHeap ライブラリの**前に**リンクしなければなりません。) |
| **msvc\vc7** | **shmfc4md.lib** | 静的リンク可能な 32 ビット版ライブラリ ファイル (Microsoft VC++ 7/MFC 4.x/5.x 用。静的リンクされた **_DEBUG** MFC のみ。Debug SmartHeap ライブラリの**前に**リンクしなければなりません。) |
| **borland** | **shdsmpbt.lib** | 32 ビット版 ランタイム SmartHeap マルチスレッド DLL インポート ライブラリ (Borland コンパイラ用。インテルのみ。) |

| ディレクトリ | ファイル | 説明 |
|---|---|---|
| **borland** | **shlsmpbt.lib** | 32 ビット版ランタイム SmartHeap マルチスレッド静的リンク ライブラリ (Borland コンパイラ用。インテルのみ。) |
| **borland** | **shsmp.lib** | 32 ビット版ランタイム SmartHeap マルチスレッド DLL インポート ライブラリ (Borland コンパイラ用。インテルのみ。) |
| **borland** | **shdw32bd.lib** | 32 ビット版デバッグ SmartHeap 静的ライブラリ (Borland コンパイラ用。インテルのみ。) |
| **borland** | **shw32d.lib** | 32 ビット版デバッグ SmartHeap マルチスレッド DLL インポート ライブラリ (Borland コンパイラ用。インテルのみ。) |
| **source** | **shmalloc.c** | ANSI C (**malloc** など) 関数の SmartHeap 定義 (カスタマイズ用) |
| **source** | **shmalmac.c** | SmartHeap ANSI C (**malloc** など) 関数のマクロ バージョン |
| **source** | **shnew.cpp** | **operator new** の SmartHeap 定義 (サポートされていないコンパイラ用にライブラリのカスタマイズやビルドを行う際に使用します。) |
| **source** | **shnewi.cpp** | **shi_New** の SmartHeap 定義 |
| **source** | **shnewhnd.cpp** | SmartHeap new ハンドラ定義 |
| **source** | **def*.c, db*.c** | 上記の **malloc** と **new** 定義で使われているデフォルト メモリ pool 定義 |
| **samples** | | Microsoft Visual C++ と Borland C++ make ファイルでの SmartHeap アプリケーションのサンプル |

# はじめに

Visual Studio .NET の誕生以来、Microsoft 社は VC++ 開発環境 (IDE) を変化させてきました。この変化に対応するために、新世代の Microsoft 開発ツールのユーザー向けに、新しいリンキング インストラクション セットが必要となりました。これを受け、ユーザーの皆様が簡単に対応いただけるように、2 つのリンク インストラクションを用意いたしました。1 つは Visual C++ 6 以前のコンパイラを使用しているユーザー向けのインストラクション、そしてもう 1 つは Visual Studio .NET 以降のユーザー向けのインストラクションです。

- Microsoft コンパイラ (Visual C++6 による) の初期のバージョンを使用している場合、この次のセクションを参照してください。
- Visual Studio.NET を使用している場合は、「Visual Studio .NET で SmartHeap/SMP を使う」を参照してください。
- 新しい SmartHeap 64 ビット ライブラリを Itanium ハードウェア上の 64 ビット Windows で使用する場合は、「Win64 で SmartHeap/SMP を使う」を参照してください。

**重要:** x86 ハードウェアで動作する 32 ビットのアプリケーションを構築する場合は、SmartHeap 32 ビット ライブラリへリンクする必要があります。Itanium1 または Itanium2 ハードウェアで動作する 64 ビットのアプリケーションを構築する場合は、SmartHeap 64 ビット ライブラリにリンクしてください。

# Microsoft Visual C++ バージョン 4、5、または 6 (Win32) で SmartHeap を使う

32 ビット Windows 版 Microsoft Visual C++ (バージョン 4、5、または 6) でアプリケーションを開発している場合は、このセクションを参照してください。

**重要:** 次のページでは、プロジェクトを SmartHeap にリンクする 2 つの方法について説明します。Microsoft Visual C++ コンパイラを使用する場合は、「クイック スタート」プロシージャをおすすめします。

クイック スタート プロシージャでリンク エラーが発生した場合や、リンク オプションを明示的に制御しなければならない場合は、「SmartHeap ランタイム ライブラリ用にアプリケーションを設定する」(ランタイム ライブラリでリンクする場合) および「SmartHeap でアプリケーションをデバッグする」(デバッグ構築の場合) で説明されている手順に従ってください。

## クイック スタート

SmartHeap をプロジェクトで使用し **`malloc`** および **`operator new`** をアプリケーションの中で置き換える場合は、このセクションの説明に従ってください。このプロシージャでは #pragma ステートメントを含むソース ファイルを使用します。#pragma ステートメントは、プロジェクトのコンパイル時間フラグをもとに、アプリケーションの正しい SmartHeap ライブラリ セットに強制的に参照されるようにします。プロジェクト設定を最初から変更する場合でも、途中で変更する場合も、正しい SmartHeap ライブラリがリンクされるため、明示的に SmartHeap ライブラリの参照を指定するよりも効果的です。

SmartHeap API を使用する場合やアプリケーションで使用する SmartHeap ライブラリを選択する場合に明示的な制御が必要な場合は、「SmartHeap ランタイム ライブラリ用にアプリケーションを設定する」および「SmartHeap でアプリケーションをデバッグする」を参照してください。

## SmartHeap ファイルの場所を指定する

- **Visual C++ で SmartHeap の include ファイル または library ファイルの場所を指定するには:**

1. Visual C++ [ツール] メニューから、[オプション] を選択します。[オプション] ダイアログ ボックスが表示されます。
2. [ディレクトリ] タブを選択します。
3. [表示するディレクトリ] で [インクルード ファイル] を選択します。
4. [追加] ボタンをクリックし、SmartHeap の **`include`** ディレクトリのパスをインクルード パス リストに追加します。

   **`c:\smrtheap\include`**
5. [追加] ボタンを再度クリックし、インクルード パス リストに SmartHeap **`msvc`** ディレクトリのパスを追加します。

   **`c:\smrtheap\include`**
6. [表示するディレクトリ] で [ライブラリ ファイル] を選択します。
7. [追加] ボタンをクリックし、SmartHeap の **`msvc`** ディレクトリのパスをライブラリ パス リストに追加します。

   **`c:\smrtheap\include`**
8. プロジェクトで MFC スタティック ライブラリを使用する場合と SmartHeap のスタティック ライブラリにリンクしている場合は、[追加] ボタンをもう一度クリックし、SmartHeap MFC 統合ライブラリにパスを追加します。デフォルトでは **`msvc`** ディレクトリのサブディレクトリです。

   例) **c:\smrtheap\msvc\vc7.**.
9. [OK] をクリックして、変更を保存します。

## Visual C++ MFC プロジェクトの設定

- **Visual C++MFC プロジェクトで SmartHeap を使用するには:**

  1. プロジェクトの stdafx.cpp ファイルを開きます。
  2. stdafx.cpp の始め、#include ステートメントの前に、次のテキストを挿入します。

     ```
     #include "shmfcsmp.cpp"
     ```

  3. stdafx.cpp の変更を保存します。
  4. [リビルド] を [ビルド] メニューから選択します。(Visual C++4.x/5.x/6.x)
  5. Release プロジェクト コンフィギュレーションおよび Debug プロジェクト コンフィギュレーションの両方に対し [リビルド] コマンドを繰り返します。

## Visual C++ 非MFCプロジェクトの設定

- **Visual C++ プロジェクトで SmartHeap を使用するには (MFC を使用しないプロジェクトの場合):**

  1. アプリケーションのプロジェクト ファイルを開きます。
  2. プロジェクトに shsmp.c ファイルを追加します。このファイルは SmartHeap msvc ディレクトリにあります。
  3. [プロジェクトの設定] ダイアログ ボックスを表示します。
  4. [設定の対象] から [Win 32 Release] を選択します(選択されていない場合)。
  5. [リンク] タブをクリックします。
  6. [オブジェクト/ライブラリ モジュール] の最初の部分、他のオブジェクト ファイルあるいはライブラリの前に次のテキストを入力します。

     ```
     .\Release\shsmp.obj
     ```

7. [設定の対象] から [Win32 Debug] を選択します。
8. [リンク] タブをクリックします。
9. [オブジェクト/ライブラリ モジュール] の最初の部分、他のオブジェクト ファイルあるいはライブラリの前に次のテキストを入力します。

   ```
   .\Debug\shsmp.obj
   ```

10. [OK] をクリックし、[プロジェクトの設定] の変更を保存します。
11. プロジェクトの Release バージョンと Debug バージョンをリビルドします。

# SmartHeap ランタイム ライブラリ (Win32) 用にアプリケーションを設定する

ここでは、SmartHeap ランタイム ライブラリを使用するためにアプリケーションを設定する手順を説明します。Debug SmartHeap でのデバッグに関しては、「SmartHeap でアプリケーションをデバッグする」を参照してください。

## SmartHeap ヘッダ ファイルをソース ファイルに追加する

**重要:** SmartHeap API を呼び出さない場合は、次のページに進んでください。**`malloc`** と **`operator new`** の置換だけに SmartHeap を使用する場合は、SmarHeap ヘッダ ファイルを追加する必要はありません。

- **SmartHeap API のコールを行う各ファイルについて:**

  - C ソース ファイルの場合、C SmartHeap API の宣言を含む **`smrtheap.h`** を含めます。

    ```
    #include <smrtheap.h>
    ```

  - C++ ソース ファイルの場合、SmartHeap バージョンの **`operator new`** の宣言と **`smrtheap.h`** を含む **`smrtheap.hpp`** を含めます。

    ```
    #include <smrtheap.h>
    ```

  いずれの場合も、**`malloc`** (例 **`stdlib.h`**) または **`new`** (例 **`new.h`** または **`afx.h`**) を宣言するコンパイラまたはライブラリ ヘッダ ファイルの**後に**、**`#include`** ディレクティブがなければなりません。

## SmartHeap ファイルの場所を指定する

- **Visual C++ で SmartHeap の include ファイル または library ファイルの場所を指定するには:**

  1. Visual C++ [ツール] から [オプション] を選択します。[オプション] ダイアログ ボックスが表示されます。
  2. [ディレクトリ] タブを選択します。
  3. [表示するディレクトリ] の [インクルード ファイル] を選択します。
  4. [追加] ボタンをクリックし、SmartHeap の **`include`** ディレクトリのパスを インクルード パス リストに追加します。

     **`c:\smrtheap\include`**

  5. [表示するディレクトリ] から [ライブラリ ファイル] をもう一度、選択します。
  6. [追加] ボタンをクリックし、SmartHeap の **`msvc`** ディレクトリのパスをライブラリ パス リストに追加します。

     **`c:\smrtheap\include`**

  7. プロジェクトで MFC スタティック ライブラリを使用していて、SmartHeap のスタティック ライブラリにリンクしている場合は、[追加] ボタンをもう一度クリックし、SmartHeap MFC 統合ライブラリにパスを追加します。デフォルトでは、msvc のサブディレクトリです。
     例) c:￥smrtheap￥msvc￥vc6
  8. [OK] を選択し、変更を保存します。

## Visual C++ プロジェクト ファイルを設定する

ここでは、Visual C++ プロジェクト ファイルで SmartHeap を使用するための設定について説明します。

**メモ** アプリケーションの各 EXE および DLL に対し次の手順を繰り返す必要があります。

➢ **Visual C++ プロジェクトで SmartHeap を使用するには:**

1. [ファイル] メニューから [ワークスペースを開く] を選択し、プロジェクト ファイルを開きます。
2. [プロジェクト] メニュー (Visual C++ 2.x) または [ビルド] メニュー (Visual C++4.x/5.x/6.x) から、[設定] を選択します。[プロジェクトの設定] ダイアログが表示されます。
3. [設定の対象] から [Win 32 Release] を選択します(選択されていない場合)。
4. [リンク] タブをクリックします。
5. リンクする SmartHeap ライブラリを指定します。[オブジェクト/ライブラリ モジュール] の**最初**の部分、Visual C++ ライブラリの**前に**適切なライブラリを入力します。

| SmartHeap DLL を次のファイルと併用する場合 | ライブラリとのリンク (表示順) |
|---|---|
| マルチスレッド アプリケーションの MFC を使用しない、 | |
| 静的にリンクされた MFC 3.0 または MFC 3.0/4.x | |
| DLL を使用する EXE または DLL | **`shdsmpmt.lib`** |
| 静的にリンクされた MFC 4.x を使用するマルチスレッド | |
| アプリケーションの EXE または DLL | **`shmfc4m.lib`**、**`shdsmpmt.lib`** |

| SmartHeap を 次のファイルに静的リンクする場合 | ライブラリとのリンク (表示順) |
|---|---|
| MFC を使用しない、または静的にリンクされた | |
| MFC 3.0 を使用するマルチスレッド アプリケーションの EXE または DLL | **shlsmpmt.lib** |
| 静的にリンクされた MFC 4.x を使用するマルチスレッド | |
| アプリケーションの EXE または DLL | **shmfc4m.lib**、**shlsmpmt.lib** |

**メモ** 前述のライブラリはファイルのインストール先ディレクトリの **msvc** サブディレクトリにあります。

MFC を使用する場合 MFC の DLL バージョンをアプリケーションで使用している場合は、SmartHeap の DLL バージョン (`shdsmpmt.lib`) を使用する必要があります。アプリケーションを Debug MFC とリンクしている場合は、Debug SmartHeap にリンクしてください。

6. [OK] を選択し、[プロジェクトの設定] の変更を保存します。
7. [ビルド] メニューから [ビルド *ファイル名*] を選択すると、Visual C++ とアプリケーションが再リンクします。

# SmartHeap (Win32) でアプリケーションをデバッグする

ここでは、Debug SmartHeap でデバッグを行うためにアプリケーションを設定する手順を説明します。

## Debug SmartHeap ヘッダ ファイルをソース ファイルに追加する

Debug SmartHeap API (**dbgMem**XXX) を呼び出す場合は、Debug SmartHeap ヘッダ ファイルを追加してアプリケーションをリコンパイルする必要があります。

**メモ** Debug SmartHeap ライブラリとリンクするだけで、SmartHeap のエラー検出機能をフルに実行することができます。さらに、アプリケーションが Visual C++ PDB デバッグ情報でコンパイルされている場合、Debug SmartHeap ではそのデバッグ情報からファイル名と行番号を取得できます。そのため、Debug SmartHeap API を呼び出さない場合は、アプリケーションを再びコンパイルする必要はありません。

アルファ ユーザーの方へ! インテル プラットフォームを使用しない場合、Debug SmartHeap では Visual C++ PDB デバッグ 情報からファイルと行に関する情報が取得できません。したがって、SmartHeap のアルファ バージョンを使用していて、エラー レポートにファイル名と行番号を追加したい場合は、`heapagnt.h` をソース ファイルに追加し、プリプロセッサ定数 `MEM_DEBUG` および `DEFINE_NEW_MACRO` を定義する必要があります。

- **Debug SmartHeap API を呼び出す各ファイルについて:**
  - **`heapagnt.h`** を含めます。これには ANSI C (**`malloc`**、**`free`** など) および C++ (**`new`** および **`delete`** オペレーターを含む) の両方のメモリ アロケーション関数の宣言が含まれています。**`#include`** ディレクティブは **`malloc`** (例 **`stdlib.h`**) または **`new`**（例 **`new.h`** または **`afx.h`**）を宣言するコンパイラまたはライブラリ ヘッダ ファイルの**後に**配置しなくてはなりません。

```
#include <heapagnt.h>
```

**メモ** 既に `smrtheap.h`、`shmalloc.h`、または `smrtheap.hpp` が含まれている場合は、`heapagnt.h` を含める必要はありません。Runtime SmartHeap ヘッダ ファイルでは MEM_DEBUG が定義されると自動的に `heapagnt.h` が追加されます。

## Debug SmartHeap ファイルの場所を指定する

- **Debug SmartHeap の include ファイルと library ファイルの場所を指定するには**

1. [ツール] から [オプション] を選択します。[オプション] ダイアログ ボックスが表示されます。
2. [ディレクトリ] タブを選択します。
3. [表示するディレクトリ] の [インクルード ファイル] を選択します。
4. [追加] ボタンをクリックし、Debug SmartHeap の **`include`** ディレクトリのパスを インクルード パス リストに追加します。

   ```
   c:\smrtheap\include
   ```

5. [表示するディレクトリ] から [ライブラリ ファイル] をもう一度、選択します。
6. [追加] ボタンをクリックし、 Debug SmartHeap **`msvc`** ディレクトリのパスをライブラリ パス リストに追加します。

   ```
   c:\smrtheap\msvc
   ```

7. プロジェクトで MFC スタティック ライブラリを使用し、SmartHeap のスタティック ライブラリにリンクしている場合は、[追加] ボタンをもう一度クリックし、SmartHeap MFC 統合ライブラリにパスを追加します。デフォルトでは、msvc のサブディレクトリです。 (例 **c:\smrtheap\msvc\vc6)**
8. [OK] をクリックして、変更を保存します。

## Visual C++ プロジェクト ファイルを設定する

ここでは、Visual C++ プロジェクト ファイルを設定して、対応する EXE または DLL を Debug Smartheap でデバッグする方法を説明します。

**メモ** アプリケーションの各 EXE および DLL に対し次の手順を繰り返す必要があります。

- **Visual C++ プロジェクトで Debug SmartHeap を使用するには:**

1. [ファイル] メニューから [ワークスペースを開く] を選択し、プロジェクト ファイルを開きます。
2. [プロジェクト] メニュー (Visual C++ 2.x) または [ビルド] メニュー (Visual C++4.x/5.x/6.x) から、[設定] を選択します。[プロジェクトの設定] ダイアログが表示されます。
3. [設定の対象] から [Win32 Debug] を選択します (選択されていない場合)。
4. [C/C++] タブをクリックし、[デバッグ情報] から [プログラムデータベースを使用] を選択します。
5. **(このステップはアルファとパワーPCを使用する場合に必要ですが、それ以外の場合はオプションです)** インテル SmartHeap を使用している場合および Debug SmartHeap API を呼び出せない場合は、次のステップに進んでください。

   [C/C++] タブの [プリプロセッサの定義] に MEM_DEBUG=1 を追加します。

   ```
   MEM_DEBUG=1
   ```

6. **(アルファおよびパワーPCのみ)** さらに、[プリプロセッサの定義] に DEFINE_NEW_MACRO=1 を追加します。

   ```
   DEFINE_NEW_MACRO=1
   ```

7. [リンク] タブをクリックします。

8. [デバッグ情報を生成する] チェックボックスをチェックします。
9. [リンク] タブで、次の表をもとに ンクする Debug SmartHeap ライブラリを指定します。[オブジェクト/ライブラリ モジュール] の**最初**の部分、Visual C++ ライブラリの**前に**、適切なライブラリを入力します。

| デバッグしている場合 | ライブラリとのリンク (表示順) |
|---|---|
| MFC を使用しない、または非 Debug MFC あるいは Debug MFC DLL を使用する EXE または DLL | **shdw32md.lib** |
| 静的にリンクされた Debug MFC 3.0 を使用する EXE あるいは DLL | **shmfc32md.lib, shdw32md.lib** |
| 静的にリンクされた Debug MFC 4.x を使用する EXE または DLL | **shmfc4md.lib4md.lib 、 shdw32md.lib** |

> **メモ** **shdw32md.lib** ファイル、**shmfc32md.lib** ファイル、および **shmfc4md.lib** ファイル は、SmartHeap をインストールしたディレクトリの **msvc** サブディレクトリにあります。

10. [OK] をクリックして、[プロジェクトの設定] の変更を保存します。

## リコンパイルするまたは再リンクする

Debug SmartHeap API を呼び出している場合や、SmartHeap のアルファ バージョンを使用している場合は、Debug SmartHeap ヘッダ ファイルを追加してアプリケーションをリコンパイルする必要があります。このセクションの前半のヘッダ ファイルの追加についての説明を参照してください。

メモ Debug SmartHeap ヘッダ ファイルを 1 つ以上のソース ファイルに含めている場合、リコンパイルのみが必要です。ただし、Debug SmartHeap では、それぞれの EXE および DLL を常に再リンクする必要があります。

- **リコンパイルするには:**

  1. [ファイル] メニューから [ワークスペースを開く] を選択し、プロジェクト ファイルを開きます（既に開いていない場合）。
  2. [プロジェクト] メニュー (Visual C++ 2.x) または [ビルド] メニュー (Visual C++4.x/5.x/6.x) から [リビルド] を選択すると、アプリケーションがコンパイルされます。

- **再リンクするには:**

  1. [ファイル] メニューから [ワークスペースを開く] を選択し、プロジェクト ファイルを開きます（開かれていない場合）。
  2. [プロジェクト] メニュー (Visual C++ 2.x) または [ビルド] メニュー (Visual C++4.x/5.x/6.x) から [ビルド *ファイル名*] を選択すると、アプリケーションが再リンクされます。