Computex

Visual C++対応 動的テストツール

CodeRecorder VC

見える衝撃

プログラムの本当の動きが見える衝撃

https://www.computex.co.jp/crvc

「普通のアプリの上をゆけ」

潜在的な問題は表面には現れません

今、動作しているプログラムは「偶然」動いているように見えるだけかもしれません。
CodeRecorder VCは、Visual C++で作成されたプログラムの動作を様々な角度から検証し、
バグの発見から品質テストまで、性能向上を図ることができる今までにない開発支援ツールです。

CodeRecorder VCは、Visual C++で開発されるWindowsアプリケーションの「開発段階」から「最終テスト」まで幅広く扱えるツールです。開発の初期段階ではプログラムの「論理的欠陥」の発見からデバッグに役立ちます。また苦労の集大成ともいえる最終テストでは、「品質評価」や「パフォーマンス・テスト」はもちろん、稀に発生する不具合の前兆を「捉え」、様々な角度から「プログラムを見る」技術 を提供します。

導入はいたって簡単！

インポート不要　オプション変更不要　ドライバの組込み不要

ワンクリックでテスト準備完了！

必要な設定は「CodeRecorder AddIn」が自動的にVisual Studioへ登録してくれます。
わずらわしいプロジェクトのインポートやオプション変更は一切ありません。

Visual Studio（Visual C++）

導入までのメカニズム

Step1 「CodeRecorder AddIn」を起動 — Visual Studioのツールバーをクリックするだけ！

Step2 AddInから「テストコードの埋め込み」 — テストを行うソース・ファイルを選択！

Step3 Visual Studioで「ビルド実行」 — いつものようにアプリケーションをビルド！

Step4 アプリケーションの実行 — 実行結果を記録、テストデータを出力！

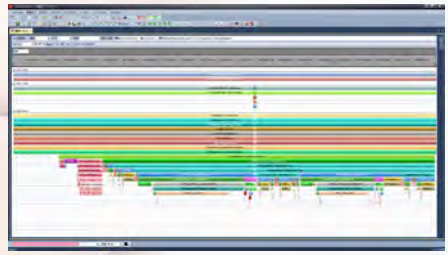
Step5 CodeRecorderで解析！ — テスト結果をレポート表示！

「なくても困らない」と思われがちなツールですが、それはちょっと誤解です。

隠れた問題を見つけたい

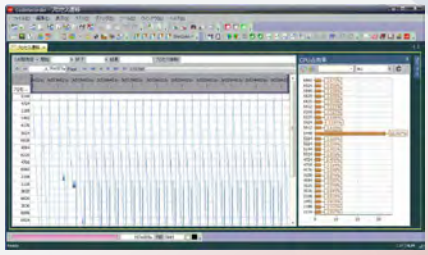
関数スタック　関数遷移　プロセス遷移

関数やスレッドの呼び出し構造を時系列で表示する、**「関数スタック」「関数遷移」「プロセス遷移」**
複雑なプログラムの流れを把握し、テスト時に発生する障害を「見える化」する衝撃のトリプル・ウィンドウ！



■関数の呼び出し経路をグラフィカルに表示
■スレッド毎にプログラム構造を表示
■コンストラクタやデストラクタを可視化
■親関数の呼び出しから再帰関数も表示可能
■対象区間のスケールを拡大/縮小が可能
■関数を色分け表示可能

■プログラムの動きと画面キャプチャ画像を連動表示
■着目位置をブック・マーク登録可能
■スレッドの切り替わりタイミングを視覚化
■ライブラリ関数対応（ユーザ・カスタマイズ可能）
■豊富な検索機能（関数、時間、サムネイル指定可能）
■指定区間を２点間計測可能

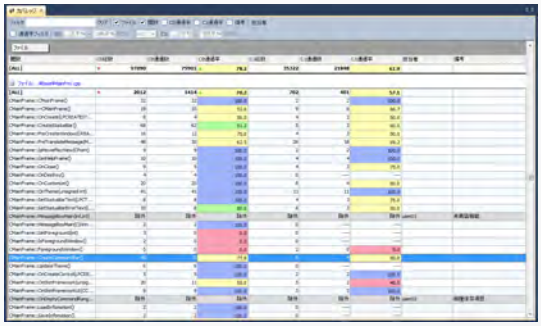


■スレッドのCPU占有率をチャート表示可能
■各スレッドにエイリアスを登録可能
■スレッドのCPU占有率を算出
■デバッグ・ポイントと連動可能
■スレッドや関数の切り替わり先にジャンプ可能

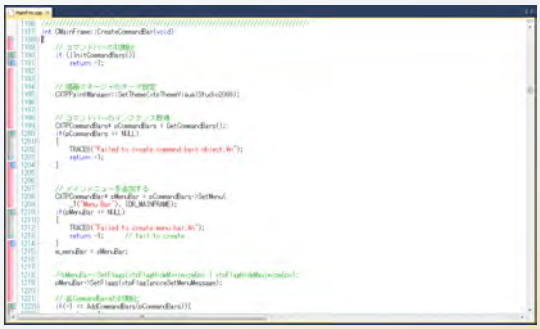
テスト漏れをなくしたい

カバレッジ　ワンパスカバレッジ　C0　C1

命令を網羅する「C0カバレッジ」、条件分岐を網羅する「C1カバレッジ」= テスト達成度を数値化する**「カバレッジ」**
テストケースの漏れをソース・レベルで表示、「実行」「未実行」箇所を確認することで、テストの「偏り」と「漏れ」を把握する！



■命令網羅（C0）対応（ステートメント・カバレッジ）
■分岐網羅（C1）対応（ブランチ・カバレッジ）
■テスト結果をマージ可能（テスター毎にカバレッジ・テスト可能）
■プログラム全体、ソース単位、関数単位での集計が可能
■テスト結果の漏れをソース・レベルで表示可能

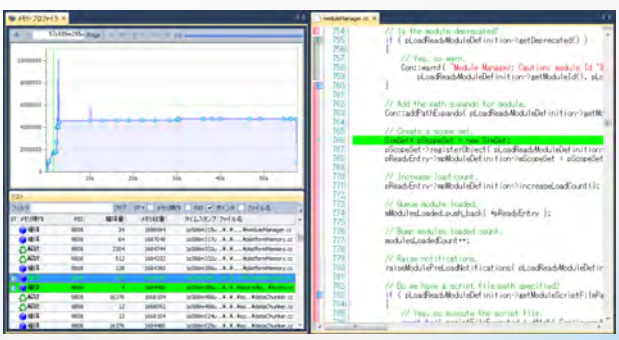


■詳細なカバレッジ除外設定（ソース、関数、指定ブロック）
■担当者、コメントなどの備考（メモ）を登録可能
■テスト結果をコンパクトにするワンパス・カバレッジ対応
■テスト達成度をレベル毎に色分け表示可能
■フィルタ機能により、除外項目を非表示に設定可能

品質を向上したい

メモリプロファイラ

動的メモリの使用状況を監視する**「メモリ・プロファイラ」**
未解放なメモリや多重解放などの異常ポイントを検出！

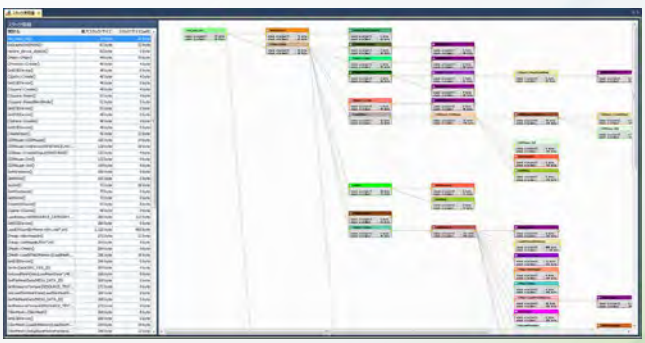


■メモリ操作関数「new/delete/malloc/free/realloc」に対応
■メモリ・リーク、多重解放などの異常ポイントをレポート表示
■異常が発生したポイントをソース・ファイルで表示
■大量のデータから異常個所をピンポイント検索

移植障害を調査したい

スタック使用量　呼び出し経路

関数の最大スタック使用量を調査する**「スタック使用量」**
限りあるメモリ資源を効率的に管理！



■最大スタック使用時の関数の呼び出し経路を追跡
■子関数の呼び出し経路とスタック使用量を算出可能
■同一関数が複数の場所からコールされた経路のスタック量を表示
■スタック積み込み順序をリストとコール・グラフで表示

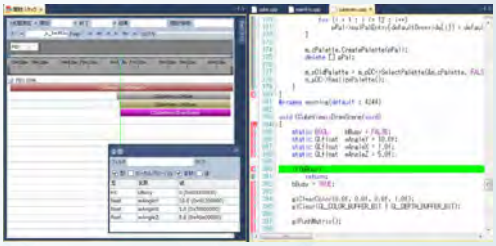
こんな経験ありませんか？

■製品リリース後に不具合が発見される
■テスト不足は否めない
■再現率の低い不具合を発見できない
■広範囲な拡張によりデグレードがないか不安

再現性の低い不具合を見つけたい

トレースバック　ブレークポイント　変数参照　画面キャプチャ

稀にしか発生しない現象を見逃さない**「トレース・バック機能」**
捉えた現象は、充実のデバッグ機能でソースライン・デバッグ可能！
キャプチャ画像とプログラムの実行軌跡を同期させることで、GUIの状態を確認しながら現象を再現することができます。



■変数の値を表示可能
■充実のデバッグ機能（実行/ブレーク/ステップ実行/カム実行）
■プログラムの動きを逆戻し可能（逆実行/逆ステップ/逆カム実行）
■ブレーク・ポイント設定可能（パス・カウント付き）
■デバッグ対象スレッドを固定可能
■オリジナル・ソース・ファイルでデバッグ可能

■テスト対象アプリケーションをキャプチャ可能
■キャプチャ画像を各ウィンドウと同期表示
■キャプチャ画像を動画として表示可能（再生/停止/一時停止）
■画像の全体、または一部分の変化率から問題箇所の検索が可能
■キャプチャ領域を指定可能

処理速度を上げたい

実行時間　周期時間　プロファイラ比較　CPU負荷率

関数ごとの「実行時間」や「呼び出し回数」、最大/最小/平均時間を計測する**「プロファイラ（実行時間/周期時間/プロファイラ比較）」**
関数から呼び出される子関数の占有時間はもちろん、関数の呼び出し経路を表示することでプログラムのボトルネックを発見！
また、プログラム実行中のCPU負荷率やメモリ使用量などをロギングする**「CPU負荷率グラフ」**を搭載。

■実行時間比率をチャート表示
■関数の呼び出し順序をフロー表示
■関数の呼び出し周期時間をレポート表示
■呼び出し回数毎に詳細レポート表示

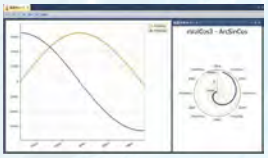
■各セッション間の相違点をレポート表示
■「向上」「低下」を５段階で評価
■各要素毎に比較結果を表示可能
■比較対象レポートを選択可能

■標準でCPU/User/System/IO/Idle時間の表示
■CPUコア毎にCPU負荷率を表示可能
■表示アイテムをカスタマイズ可能
■テンポラリ・マーカの登録

変数のバラツキを見極めたい

変数チャート

不具合の原因を未然に防ぐ**「変数チャート」**
変数値の変化をグラフ化し、突出箇所を検出！

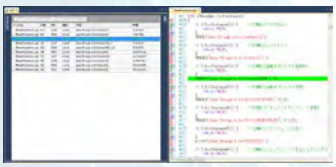


■変数値の変化を分布化する、レーダ・チャート表示
■検索機能により、基準値「以上」「以下」を検出可能
■時間、データ軸でのスケール変更可能
■複数の変数を表示可能

ログを確認したい

ログ

ユーザ指定の特殊情報をアウトプット可能な**「ログ」**
Printfデバッグと同様に任意の文字列をテスト結果に出力！

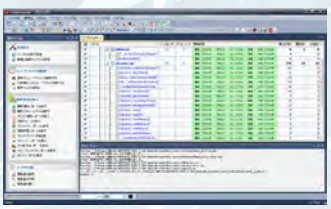


■printf、TRACEマクロ対応
■Release版でもTRACE文の結果をログとして出力
■高度なフィルタ機能で、大量のログからピンポイント検索
■ログの出力箇所をソース・ファイルで確認可能
■大量のログをストレスなく表示する高速設計

テスト・コードを厳選したい

プロジェクト

テストコードの埋め込みを高度にカスタマイズする**「プロジェクト」**
テスト箇所を厳選し、クリティカルな処理をテスト！



■ユーザ指定可能なフォルダ管理
■関数毎に埋め込みレベルを設定可能
■ローカル変数の埋め込み有無を指定可能
■Visual Studioに登録されているソース・ファイルを自動検出
■コンパイラ・オプション、プリプロセッサを自動登録

報告書を提出したい

報告書

テスト結果を提出用レポートに整形する**「報告書」**
単調になりやすいテスト結果を「見やすく」ブラウザ形式で作成！

■カバレッジ、プロファイラ結果に対応
■表紙、ヘッダ、フッタ対応
■プロファイラ結果をチャート図で出力可能
■ブラウザ上で最終確認用のチェック・リストを設定可能
■CSV出力で外部アプリケーションでのレポート編集が可能

あなたに「必要」なこと

動的テスト・ツールを上手に使いこなすポイントは、開発対象となるアプリケーションの「どこに技術的難易度があるのか？」を把握し、使用する機能を選別することが重要です。やみくもにCodeRecorderから提供される計測値を確認しても、「結果ばかり押しつけられる」と感じるかもしれません。

あなたが開発したアプリケーション
「どこが複雑なのか？」教えてください。

「そこに」必ず問題解決への道筋が「見えて」きます。

「思い込み」の壁

関数遷移 | 関数スタック | プロセス遷移

(隠れた問題を見つけたい)

Windowsアプリケーションの内部では、イベントやスレッドなど実に複雑に動作しています。　また、外部ライブラリを利用することで複雑化と大規模化が進み、アプリケーション内部の詳細を把握することが非常に難しくなっています。「プログラムは正常に動作している！」一見、正常に動作しているプログラムですが、実は問題を抱えていることが「よく」あります。思い通りに動作している時ほど、隠れた問題に気づかないのです。

関数、スレッド、コンストラクタ、デストラクタ、再帰処理 を「見える化」！

「テスト不足」自覚症状のないアナタ

カバレッジ | ワンパスカバレッジ | C0 | C1

(テスト漏れをなくしたい)

アプリケーションの品質評価には、さまざまなテスト・ケースが考慮された検査項目があります。
検査項目に従いテストを実施し、品質評価試験に合格したにもかかわらず、時に欠陥が発見されることがあります。
現在のテスト内容に「偏り」「漏れ」があるのです。

明確なテスト達成度を数値で指標化！

「アクセス違反」見たくなかった、その呪文

トレースバック | ブレークポイント | 変数参照

(再現率の低い不具合を見つけたい)

デバッグも終わりRelease版で最終テスト！「アクセス違反が発生しました」のメッセージが・・・ せっかく発見した不具合ですが、Debug版で調査したところ「現象が出ない・・・」錯覚だったのでしょうか？
違います！　メモリの初期化状態やタイミングなど、現象が発生する事象が変わったのです。

解決すべき問題究明のチャンスを見逃さない！

「デグレード」必殺技ではありません

実行時間 | 周期時間 | プロファイラ比較 | 報告書 | CPU負荷率

(処理速度を上げたい)

機能拡張やセキュリティ向上など日々アップグレードされる製品。
単体テストも結合テストも行ったはずなのに
「バージョンアップしたら遅くなった！」 のクレームが・・・
担当機能の改善という見地でプログラムを変更すると、問題をひきおこすことがあるのです。

各セッション間における相違点を
パフォーマンスの「低下」と「向上」でレポート！

「メモリ・リーク」恐れる者ほど救われる

メモリプロファイラ

(品質を向上したい)

プログラムにおけるメモリ・リークは非常に見つけにくい現象の一つです。
リーク・サイズが少量でも、リーク頻度が加速すれば、プログラムのパフォーマンスは段階的に劣化し、メモリ不足による「さまざまな症状」が現れます。
こうなると問題の原因がどこにあるのか判断するのは困難です

未解放なメモリや多重解放などの異常ポイントを検出！
new/delete/malloc/free/reallocに対応

「ポータビリティ」ご利用は計画的に！

スタック使用量 | 呼び出し経路

(移植先での障害を調査したい)

Windowsアプリケーションの時に気にしなくてもよかったスタック・サイズですが、移植先のモバイル端末ではメモリ容量が制限されるため、無駄にメモリを消費するアプリケーションは「悪」になりかねません。
最悪システム・ダウンを引き起こす可能性もあります。

限りあるメモリ資源を効率的に管理する！

「画面キャプチャ」あの場所に連れて行って (GUIの状態を監視したい)

画面キャプチャ

画面キャプチャとプログラムの動きを同時に記録することで、表示系のバグからプログラムの問題箇所が丸裸！

その理由は？

「画面キャプチャ」で見逃さない！
×
「関数スタック」で見える化！
＝「プログラムを見る」技術の神髄！

サムネイル表示
キャプチャされた画面をサムネイル表示することで、GUIの状態とプログラムの動きを相対的に検証することが可能になります。

関数スタック表示
画面がキャプチャされた時点のプログラムの動きをスタック形式で表示します。プログラムの論理的な呼び出し経路を確認するのに便利です。

検索ペイン
検出画像を一覧表示し、基準画像との差分が強調表示されます。
動画中から目的となる画像にピンポイント移動することで、問題箇所となる場所を即座に把握できます。

ソースファイル表示
現在のタイムラインからプログラムの動きをソース・ファイルで確認できます。ソース・レベル・デバッグも可能！

動画再生/検索
キャプチャ画面を動画として再生することはもちろん、基準画像からの変化率で目的の画像を動画中の全フレームから検索することが可能です、

タイムライン・ペイン
キャプチャされた動画をフレーム単位で時系列に表示します。
動画をフレーム単位で詳細に確認するのに便利です。

動作環境

CodeRecorder VCは、以下の環境でご利用頂くことを推奨しております。

パソコン	推奨CPU	Intel Core i5プロセッサー 以上	
	推奨メモリ	4GB以上	
HDD		データ取得用に数十GB～数百GB の空きがあることが望ましい	
OS		Windows Vista	32ビット版
		Windows 7	32ビット版、64ビット版
		Windows 8	日本語OSのみ対応
		Windows 8.1	
		Windows 10	
対応言語 Visual Studio		Visual C++ 2005	CodeRecorder AddIn ソフト使用時 ※1
		Visual C++ 2008	
		Visual C++ 2010	
		Visual C++ 2012	
		Visual C++ 2013	
		Visual C++ 2015	
		Visual C++ 2017	
対応プラット・フォーム		Win32	Visual Studio で作成されたWin32 アプリケーション※2
対応言語		C/C++言語、Microsoft Visual Studio 上で動作するVisual C++に準じる	

※1： Visual Studio Express はCodeRecorder AddIn 機能を利用できません。
Visual Studio Express をご使用の場合は、CodeRecorder VC のプロジェクト･ウィンドウから
ソースファイルへテスト・コードの埋め込みが可能です。
※2： 64ビットアプリケーション（Win64）対応予定。

製品構成

■ソフトウェア “CodeRecorder VC” (CD-ROM)
■USBドングル (CodeRecorder VC用ライセンス)

ライセンスについて

導入スタイルにあわせて、ライセンス形態をご選択ください。

USBドングル・ライセンス

1ライセンスにつき1ユーザのみ使用可能なライセンス形態です。

- 複数のPCへCodeRecorder VCをインストール可能
- USBドングルを挿入したPCでのみ起動可能

CodeRecorder VCを多数導入されるお客様向けのライセンス形態もご用意しています。
詳しくは当社営業部までご相談ください。

FREE TRIAL

30日間試用版のダウンロード

コンピューテックスでは、より多くの方に動的テストツール「CodeRecorder VC」を知っていただくために、試用版のダウンロードを実施しております。CodeRecorder VC 初回起動後30日間は、製品版と同等のすべての機能が試用いただけます。

是非この機会に、動的テストツール CodeRecorder VCに触れてみてください！

30days FREE TRIAL

お申し込みはWEBで！

CodeRecorder VC 試用版ダウンロード　Search

URL: https://www.computex.co.jp/crvc

Computex®
株式会社コンピューテックス

営業部
TEL：075-551-0528(代) FAX：075-551-2585
E-MAIL：sales@computex.co.jp
https://www.computex.co.jp/

CK0164(C)1709