各種のプログラミングツール

エディタ	• コンパイルとエラー検出をエディタ内で行う. • プログラムの圧縮表示 • 編集している言語に応じた会話型ヘルプ • 一般的な言語機能のテンプレート • 自動インデンテーション • マクロ言語のサポート • 正規表現を用いた検索と置換 • 複数ファイルの同時編集 • 複数レベルの undo
ファイル比較ツール	• 編集前後のファイルを比較することで，修正忘れを防止する. • プログラムの変更履歴を管理する.
ソースコード清書ツール（ビューティファイア）	• ソースコードのインデントを一定の規則に合わせることで，プログラミングスタイルを統一する. • 誤ったインデントを未然に防止する.
ブラウザ	プログラムを圧縮表示することで，プログラミングスタイルの不一致を把握する.
クロスリファレンスツール	• ルーチン間の呼び出し関係を一覧表示する. • ルーチンと大域変数の参照関係を一覧表示する. • これらの情報を用いてモジュール化をより適切に行える.
ソースコードチェッカ	• ソースプログラムの文法チェックおよび意味的チェックをコンパイラよりも厳格に行う.（代表例：lint） • プログラミングの誤りを事前に検出することで，テストの手間を減らす.
メトリクスリポータ	ソースプログラムを分析して，各種のメトリクス値を計量する．メトリクス値の高い箇所は修正すべき箇所の候補となる. **メトリクス**：ソフトウエアの複雑さを図るための客観的尺度。コード行数、Function Point 値をはじめとして、多数のメトリクスがある。
コンパイラ・コンパイラ（コードジェネレータ）	• コンパイラ・コンパイラはプログラム言語の文法を入力すると，当該言語のプログラムを構文解析するプログラムを生成するツール. • 代表例：yacc, bison, JavaCC
リンカ	• 複数のプログラミング言語で書かれたプログラムを単一のシステムとして結合する. • モジュール毎に最も適した言語での実現が可能になる.
ライブラリ	• ライブラリ部分はプログラミング，テストが不要になる．多くの場合，ライブラリ購入コストは自力開発コストより低い. • 例：キーボード入力，マウス入力，GUI 作成，画面出力とプリンタ出力，グラフィックス，マルチメディア作成，データベース操作，通信とネットワーキング，テキスト処理，数学的処理，ソーティング，データ圧縮，コンパイラ作成
プリプロセッサ	マクロ展開によるソースコードの置換を活用すると，ソースコードのコード量削減，複数モジュールの統合などが可能になる.
プロファイラ	• 各文の実行回数を計測する. • ルーチン/文の実行時間を計測する. • これらを通じて，実行時間の集中部分を把握するための情報を提供する.