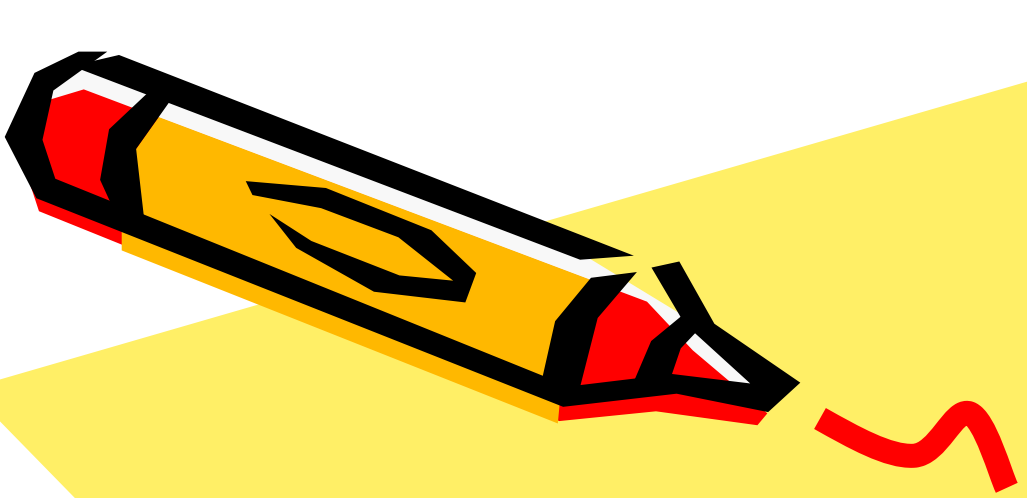
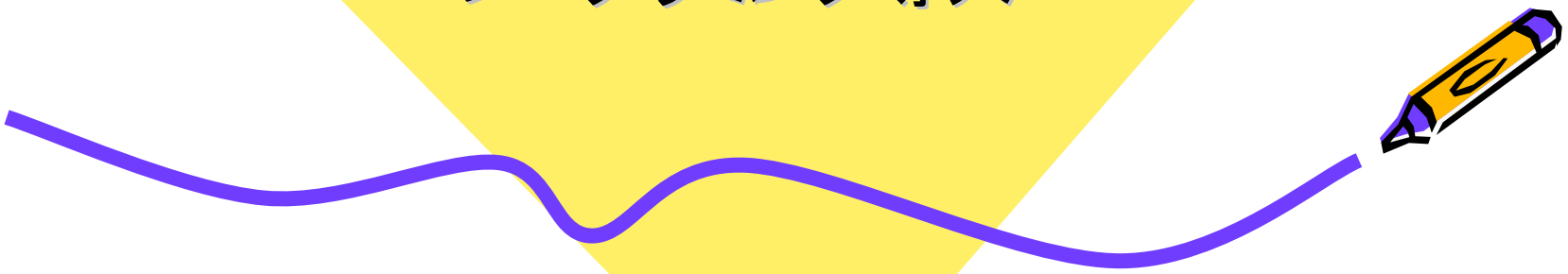


コンピュータ概論　５回

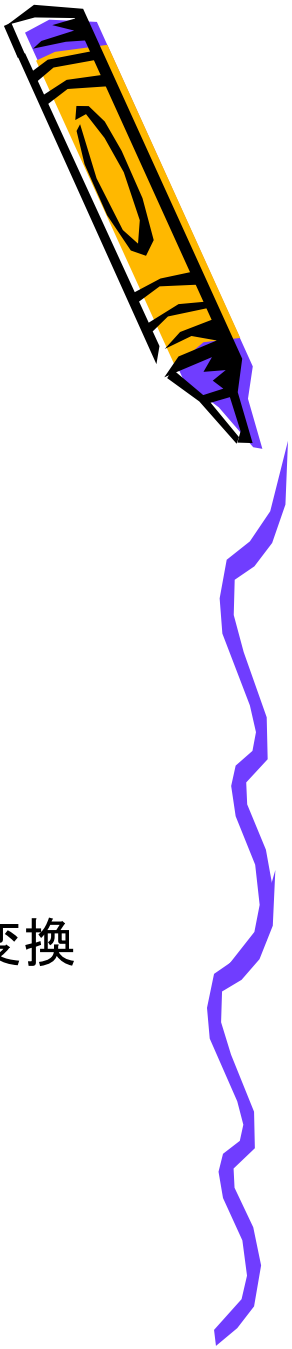
プログラミング導入



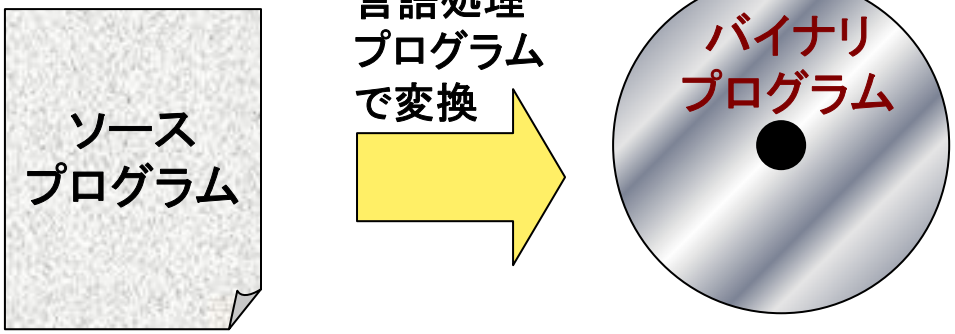
おさらい：プログラム

- コンピュータを動作させる命令
  - 処理装置の動作命令の集まり
  - 動作手順も規定（program＝計画表）
  - 処理装置は命令を逐次（1つ1つ順番に）実行
- プログラムが無ければコンピュータは動作しない
  - プログラムされていない動作は出来ない

プログラミング言語



- プログラムを記述するための人工言語
  - 機械語
    - 処理装置の動作命令プログラムそのもの
    - 処理装置の動作原理の詳細を理解していないと記述できない
    - バイナリプログラム
  - プログラミング言語
    - 人間の考える概念を記述しやすく設計
    - 言語処理プログラム（OSの構成要素）を用いて機械語に変換
    - ソースプログラム

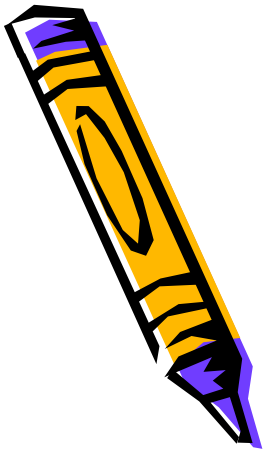


言語処理プログラム

プログラミング言語で記述されたソースプログラムを
バイナリプログラムに変換し、実行可能にするための
OSプログラム群（OSに含まれる）

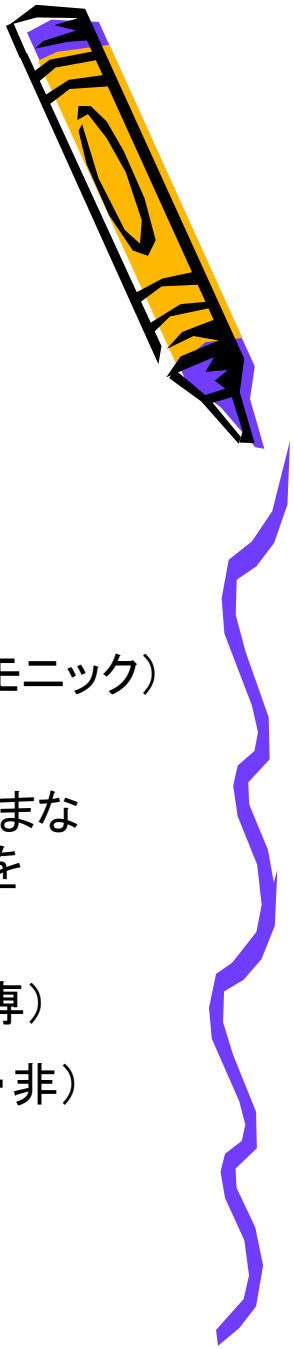
- 別名言語プロセッサ
- プログラム開発には必須
  - コンパイラ
    - 記述したプログラムを機械語に翻訳
  - アセンブラ
    - アセンブリ言語を機械語に変換
  - リンカ
    - OSの各種モジュール（ライブラリ）とプログラムを連結
  - インタプリタ
    - ソースプログラムを一命令ずつ解釈しながら実行

言語処理プログラム



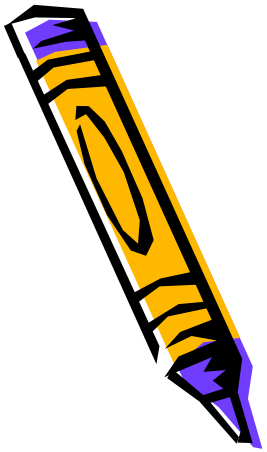
- 言語処理プログラムの分離
  - 昔　コンピュータは専門家だけのもの
  - 今　「コンピュータ利用者≠ソフトウエア開発者」
    - ほとんどのOSでは言語処理プログラムの大部分（特にコンパイラ）を分離し、アプリケーションとして別売り
  - フリーの言語処理プログラムの登場
    - GNUプロジェクト
- 分離によるメリット
  - OSの低価格化（一般利用者のメリット）
    - 一般利用者は、よりコンピュータの導入がし易くなった
  - プログラム開発環境の充実（専門家のメリット）
    - アプリケーション商品となったことで市場原理が働き、高機能化、低価格化などの方向で多彩な製品が提供

プログラミング言語

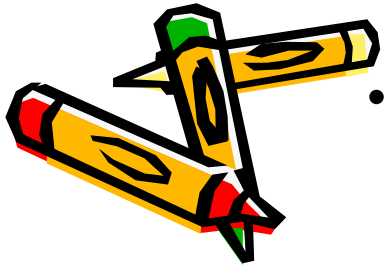


- 代表的な言語
  - **C** UNIX OSを記述するために作られた言語（高・汎・手）
  - **Fortran** 主に科学技術計算用に作られた言語（高・汎・手）
  - **PASCAL** アルゴリズムの教育用に作られた言語（高・汎・手）
  - **アセンブリ言語** CPU固有の機械語を人間にわかりやすい記述（ニーモニック）で表記できる言語（低・汎・手）
  - **Java** オブジェクト指向という概念に基づいて作られ、さまざまなOSでの利用やネットワーク（インターネット）での利用をはじめから想定して作られた言語（高・汎・手）
  - **SQL** データベースの操作をするために作られた言語（高・専）
  - **Prolog** 論理数学に基づいて作られた人工知能向け言語（高・非）

プログラミング言語の分類



- 高級・低級
  - 低級言語
    - 処理装置の動作原理を考慮しないと記述できない言語
    - 処理装置を効率良く動作させることができる
    - 処理装置に精通していないと使えない
    - 機械語、アセンブリ言語など
  - 高級言語
    - 人間の考えるアルゴリズム（プログラムの論理的な枠組）を表現するのに向いている言語
    - 論理的な枠組みが出来れば、コンピュータに詳しくなくても使える
    - 言語処理プログラムで変換するため、処理装置の動作が効率的でない
    - 一般的なプログラミング言語



機械語とプログラミング言語の比較

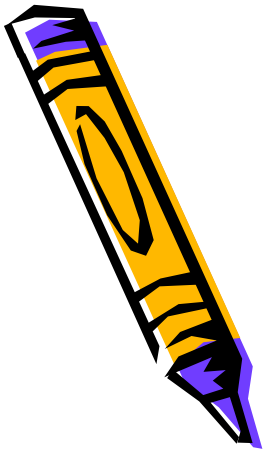
- 機械語（アセンブリ言語）
  - 無駄なくCPUを動作させることが可能
  - プログラムにミスがあってもCPUは記述どおりに動作するため、暴走することがある
  - CPU固有の記述のため、他のCPUでは使うことができない
  - CPUの動作を理解しなければならないため、専門家でないと記述が難しい
- 高級プログラミング言語
  - コンパイルによる自動翻訳を行うため、無駄な機械語の命令や動作が発生
  - コンパイル時に内容のチェックが行われるため、比較的ミスを発見しやすい
  - CPUとは独立した記述のため、コンパイラさえ用意できれば、さまざまなCPUで利用できる
  - 人間の思考に近い言語であるため、比較的一般の利用者にも記述できる

※近年のCPUの動作速度の向上やメモリの大容量化などにより
機械語でプログラムを作らなくても充分高速に動作できるようになったが、
機械制御などの専用CPU（組込みCPU）では、動作速度やメモリが充分
ではないので、現在でも機械語やアセンブラが用いられている。

プログラミング言語の分類

- 専用と汎用（特化と汎化）
  - 専用言語
    - ある特別な種類の問題（例えば、建築の構造計算や数値処理）を扱いやすくして、その他の用途への使用を想定していない言語
    - その分野の専門的な知識があれば使える
    - SQL（データベースアクセス用）やS（統計処理用）など
  - 汎用言語
    - さまざまな問題解決に利用が可能な言語
    - コンピュータの動作そのものに関する処理が可能
    - 専用言語で実現可能なことは全て実現可能だがコンピュータ自体の知識も必要
    - 一般的なプログラミング言語

プログラミング言語の分類



- 手続き型と非手続き型
  - 手続き型言語
    - 問題を解決する手順をキチンと記述する言語
    - 解き方が判っている問題に有効で、一般的に非手続き型よりも、高速に処理できる
    - 機械語を含む一般的なプログラミング言語
  - 非手続き型言語
    - 解くべき問題だけを記述して解決までの手順はコンピュータ任せとする言語
    - 解き方の手順が判明していない問題を扱うことができる
    - 一般的にはあまり使われていない
    - Prolog、LISPなどの人工知能の分野で主に使われる言語

C

- 通称：C言語
- 授業で使用するプログラミング言語
- 1972　Dennis M. Ritchieが設計
- UNIX OSを作るために作られた汎用言語
- 高級言語だが、低級記述も可能
- 世界中で現在も広く利用

Cプログラムの例

```
#include<stdio.h>
main(void){
  printf("Hello, world.¥n");
}
```

画面上にHello, world.と表示する。

Cプログラムの例

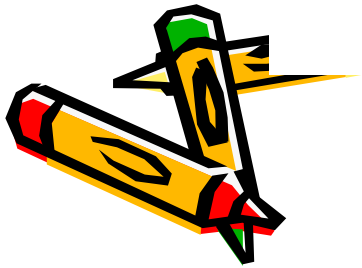
```
#include<stdio.h>
```

プリプロセッサ

```
main(void){
  printf("Hello, world.¥n");
}
```

関数

- main:関数名
- void:仮引数列(何も無いという意味)
- {…}の中:処理内容
  - 文字列“Hello world.”(改行)を表示

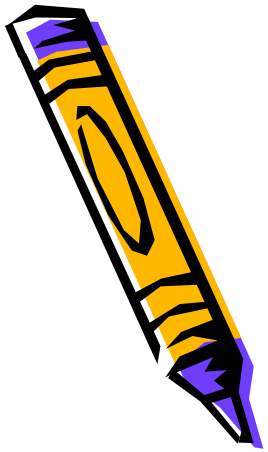


Cプログラムの構造（文法）

- プログラム
  - 関数の集合（少なくとも1つ）
- プリプロセッサ
  - 前処理用の記述（第12回で説明）
  - 一種のおまじないとして覚えておく
- 関数の書式

  関数名（*仮引数列*）{
  処理内容（複数記述可能）
  }
- 特別な関数　main関数
  - プログラム中に必ず必要
  - 授業では主に1つだけの関数（main関数）を持つプログラムを扱う

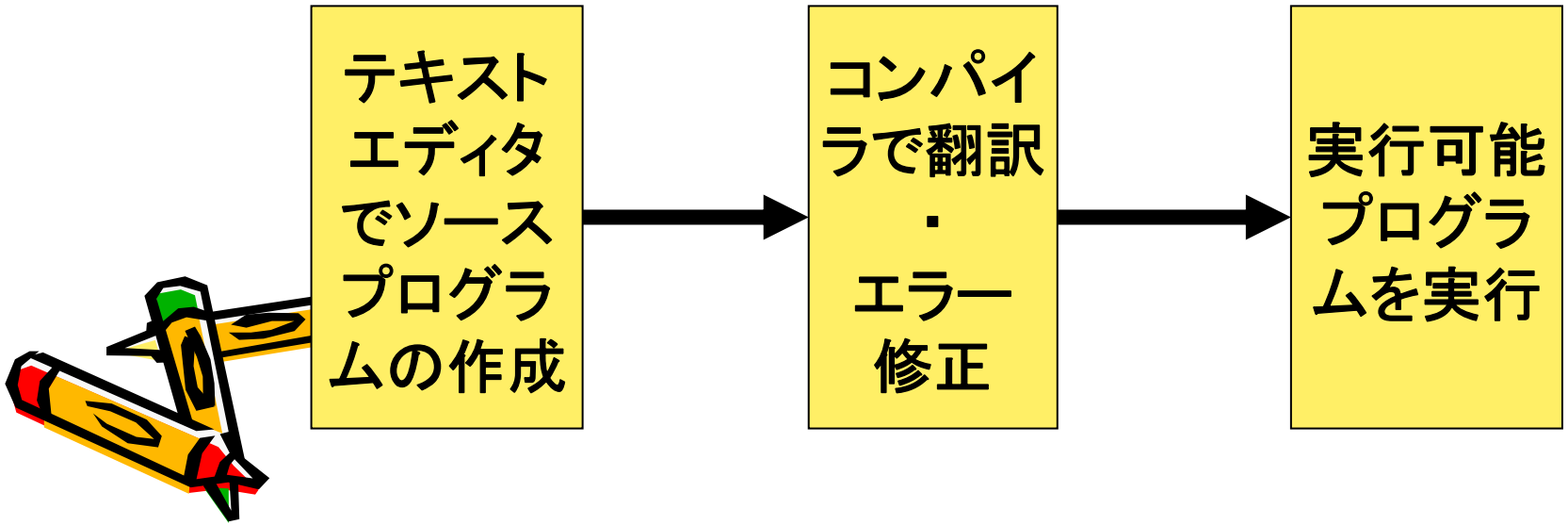
関数



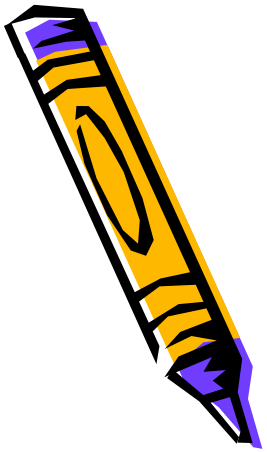
- Cで実行される処理を一塊にまとめたもの
  - まとめた処理を一気に実行することが可能
  - コンピュータに実行させる1つの命令とみなすことができる
- 種類
  - システム関数　　システムが既に用意している関数（ライブラリ関数、API）
  - 自作関数　　自分で内容を定義した関数
- 詳しくは第12回

プログラム作成から実行まで（実習開始）

- 使用する言語プロセッサ
  - Microsoft社のVisual Studio 2005 C++
  - コンパイラ・リンカのみ使用
    （Visual Studio開発環境は使わない）

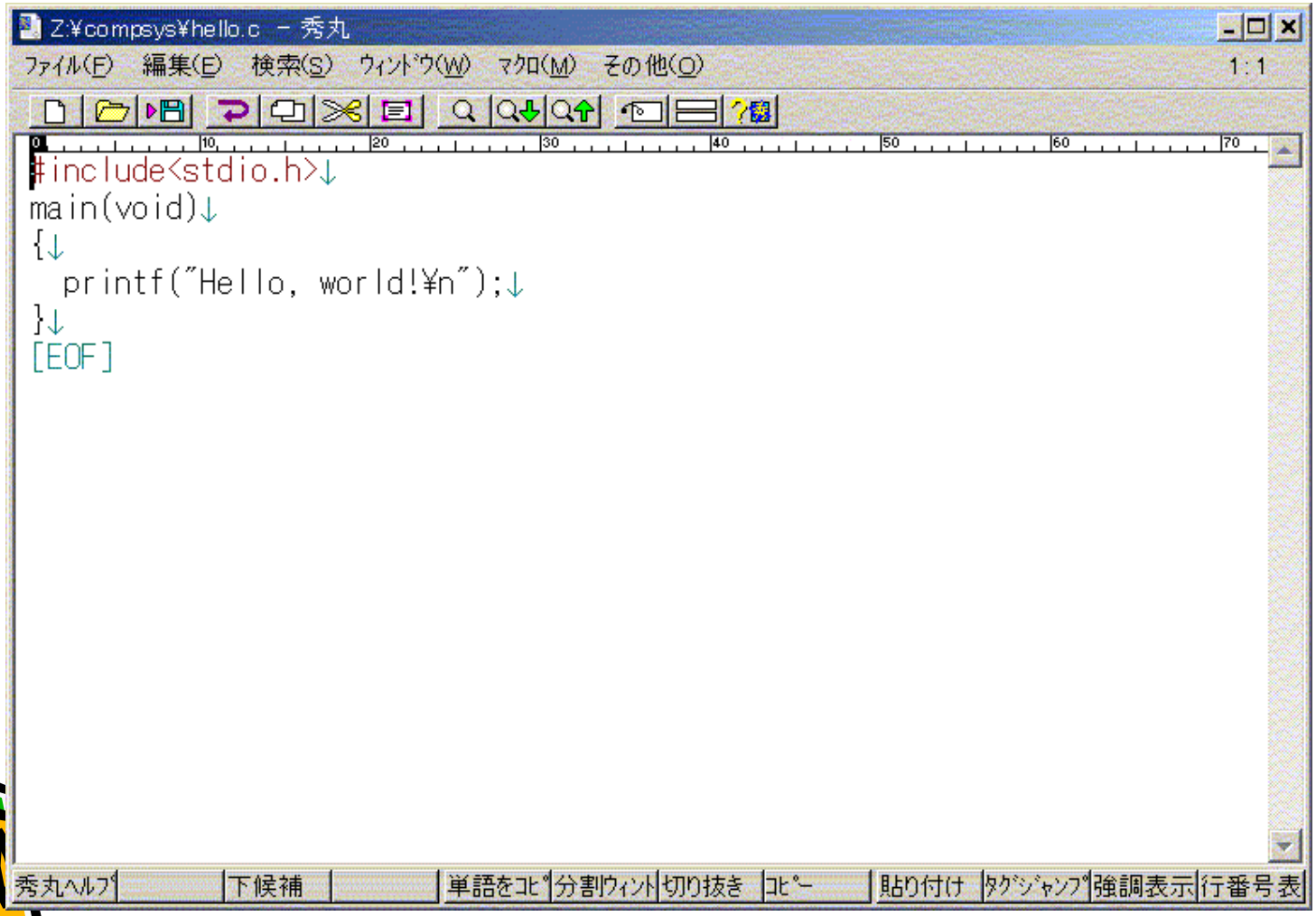


プログラムの作成



- テキストエディタを用いて作成
  - 授業では「秀丸」を使用
  - メモ帳でも可能
- ファイル名は、半角英数文字だけで命名
- ファイル名には、拡張子.cを付ける。
- 保存には、自分のホームフォルダ(Z:)に専用のフォルダを用意

秀丸エディタ

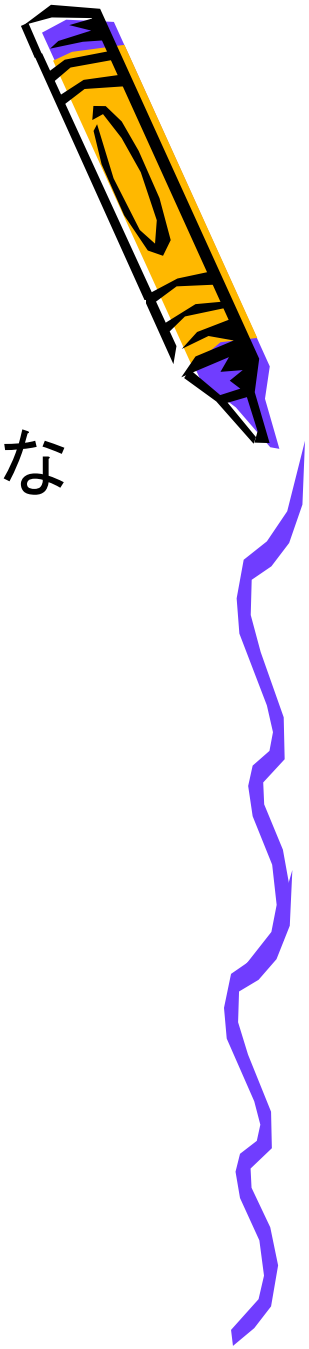

Z:¥compsys¥hello.c - 秀丸
ファイル(F) 編集(E) 検索(S) ウィンドウ(W) マクロ(M) その他(O)
1:1
#include<stdio.h>↓
main(void)↓
{↓
printf("Hello, world!¥n");↓
}↓
[EOF]
秀丸ヘルプ
下候補
単語をコピー
分割ウィント
切り抜き
コピー
貼り付け
タグジャンプ
強調表示
行番号表

実習1

- 次のプログラムを入力･コンパイル･実行せよ

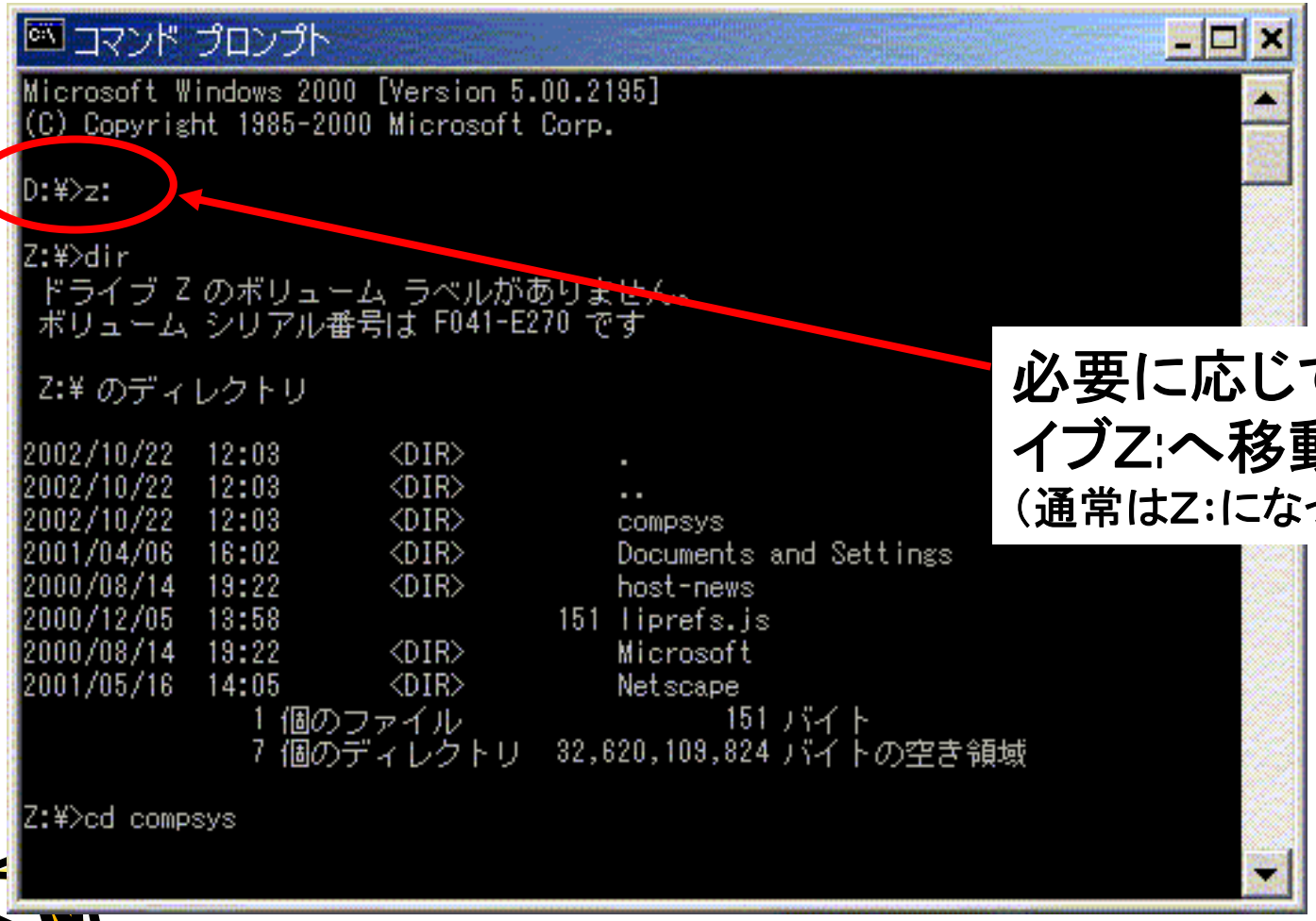
```
#include<stdio.h>
main(void){
  printf("Hello, 自分の名前¥n");
}
```

プログラムのコンパイル



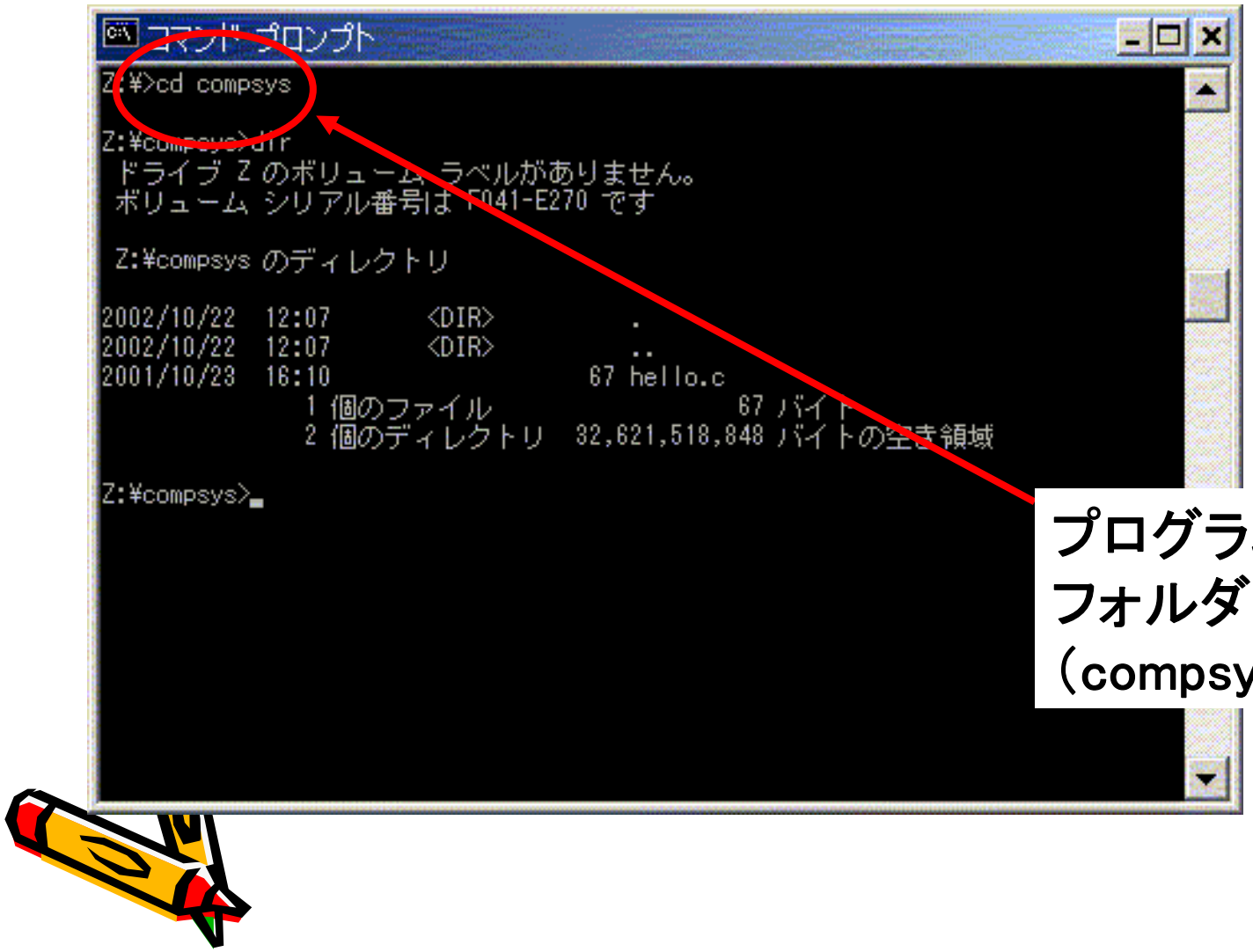
- コンパイル
  - 記述したプログラムをコンパイラで実行可能なプログラムに変換(翻訳)すること
- msvcclコマンドを使ってコンパイル
  - コマンドプロンプトで実行
  - 関連するコマンド
    - cd　フォルダの移動
    - dir　フォルダ内のファイル一覧
- コンパイルエラーが発生したら再編集

コマンドプロンプト



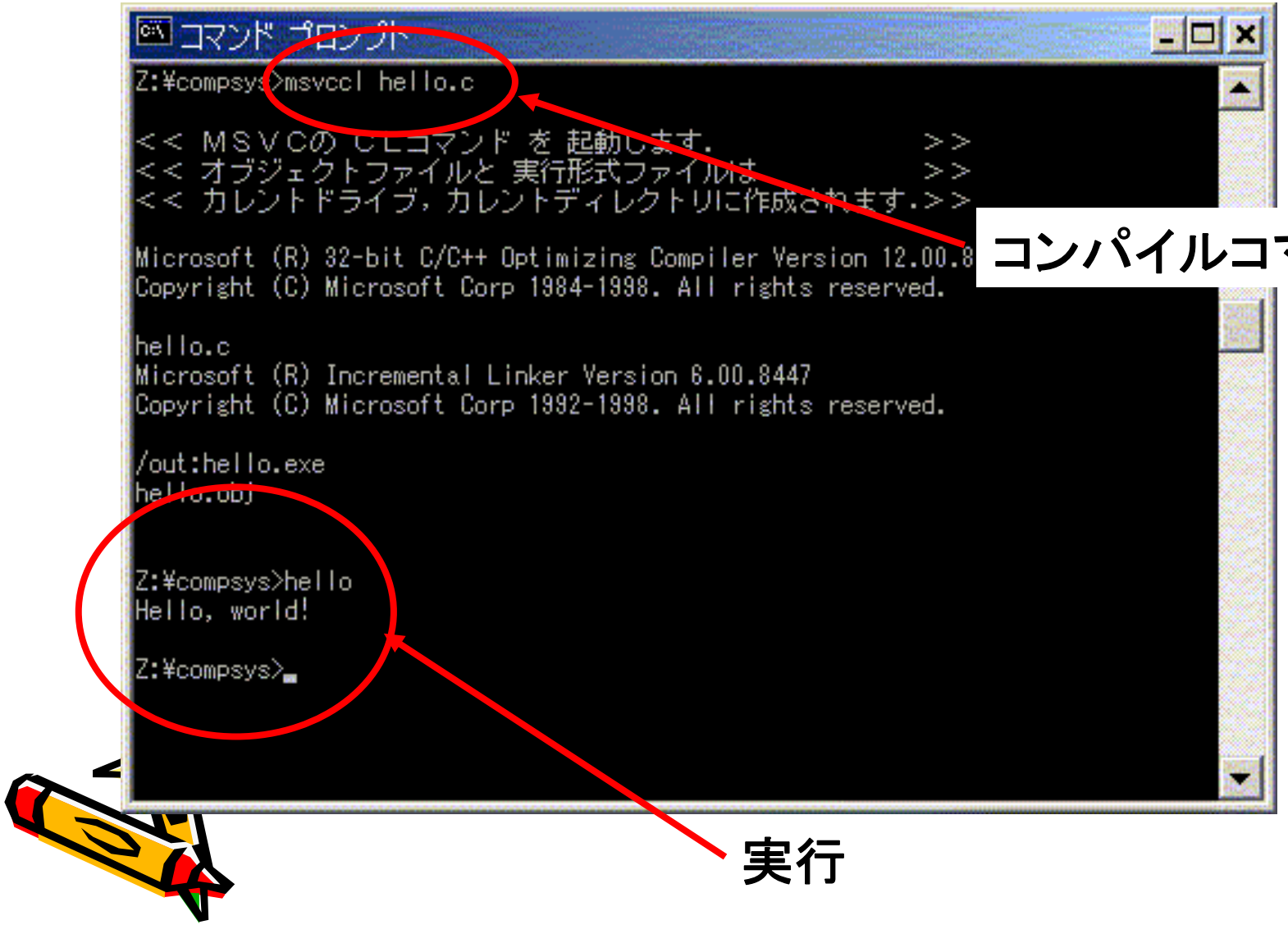
必要に応じて、ドライブZ:へ移動
（通常はZ:になっている）

フォルダの移動

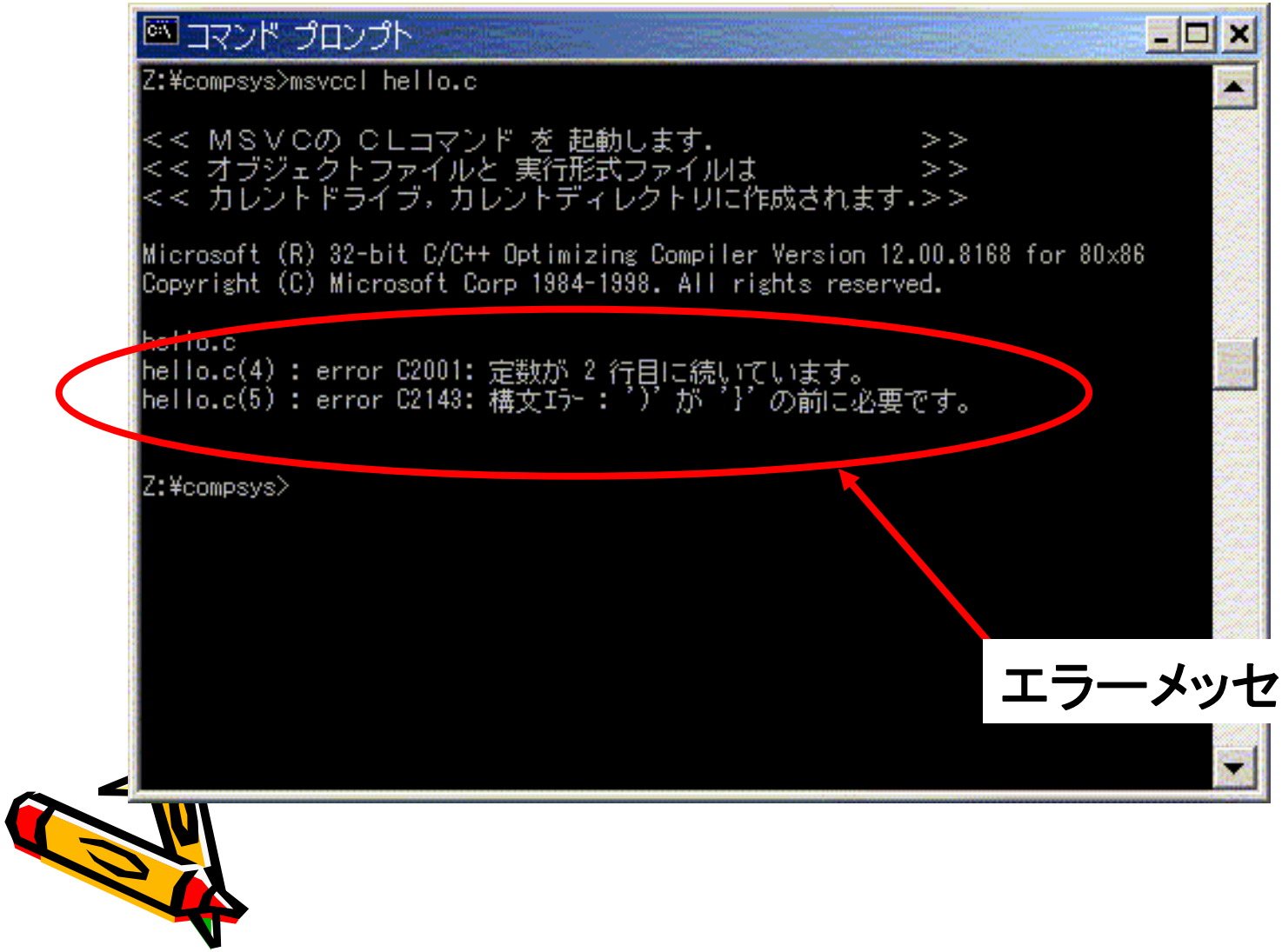


プログラム保存用
フォルダへ移動
（compsys）

コンパイル


コマンド プロンプト
Z:¥compsys>msvccl hello.c
<< ＭＳＶＣの ＣＬコマンド を 起動します. >>
<< オブジェクトファイルと 実行形式ファイルは >>
<< カレントドライブ，カレントディレクトリに作成されます. >>
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.
hello.c
Microsoft (R) Incremental Linker Version 6.00.8447
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
/out:hello.exe
hello.obj
Z:¥compsys>hello
Hello, world!
Z:¥compsys>
コンパイルコマンド
実行

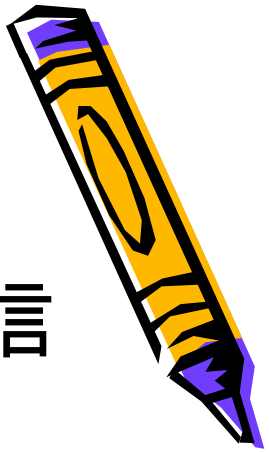
コンパイルエラー


コマンド プロンプト
Z:¥compsys>msvccl hello.c
<< MSVCの CLコマンド を 起動します. >>
<< オブジェクトファイルと 実行形式ファイルは >>
<< カレントドライブ，カレントディレクトリに作成されます.>>
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.
hello.c
hello.c(4) : error C2001: 定数が 2 行目に続いています。
hello.c(5) : error C2143: 構文ｴﾗｰ : ')' が '}' の前に必要です。
Z:¥compsys>
エラーメッセージ例

プログラムの実行

- コンパイル後に実行可能ファイル（拡張子.exe）が出来る（test.cならtest.exe）
- コマンドプロンプトで、実行可能ファイルの名前をコマンドと同じように入力すると実行
- 正しい動作が行われれば完成
- 変な動作やエラーが発生したら、プログラムの再編集

プログラムの提出



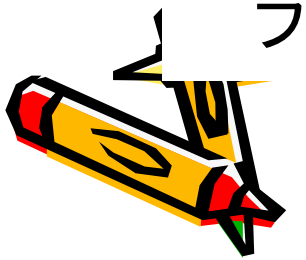
- メールに添付して自動受信システムへ送信
  - 件名は出題日を4桁の半角数字で
  - 提出先メールアドレスは授業ページで各自確認
  - 本文の書き方については次のスライドを参照
- 返信メールを受信し内容確認
  - コンパイルが成功したプログラムでも受信エラーとなる場合がある
  - 必要に応じて、再送信

本文の書き方

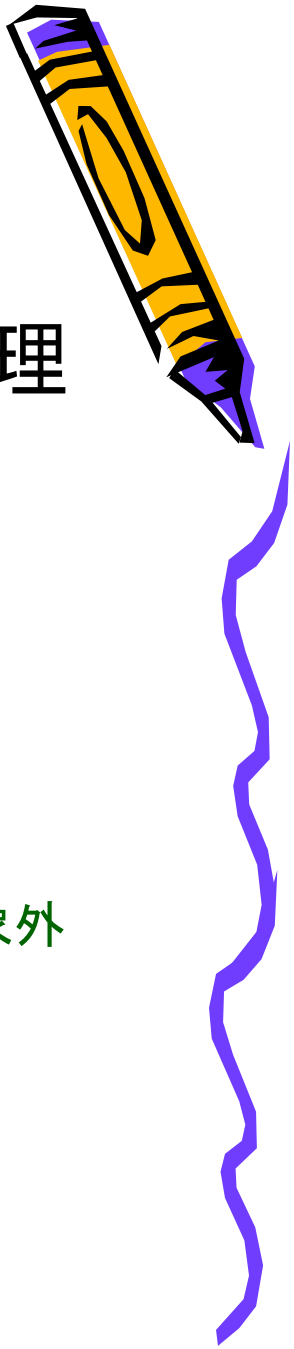
本文は、以下のように書くこと

```
/* file ファイル名 */
/* user ユーザ名 */
```

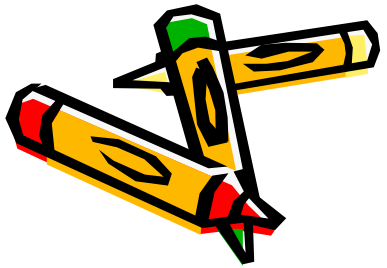
- ファイル名は実際のファイル名と異なっても良い。
- 日付とファイル名が同じ課題を提出した場合、最新の提出情報しか残らない。(過去の提出記録は上書きされる)
- ユーザ名、ファイル名は半角英数文字だけとし、ファイル名には拡張子『.c』を必ず付ける。



自動受信システム



- メール提出されたプログラムを自動処理
  - ユーザごとにファイルを保存
    - ファイル名は、コメントと日付から生成
      - 同じ日付、同じ名前で提出すると上書きされる
    - Webページで提出日などを確認可能
      - ファイルの一部のみ見ることができる
      - 採点済み、再提出指示などがチェック可能
      - 課題の答えになっていないプログラムは採点対象外
  - コンパイル時のエラーをチェック
    - 正しく実行できるかは未チェック
      - 文法上のチェックのみ
      - 後日の採点で再提出になる可能性もある



実習２

- 実習1のプログラムを提出せよ
  - 日付は今日の日付
  - ファイル名はhello2.c
  - 必ず自動受信システムの返信メールを読む
  - 返信メールがエラーの場合は、再送信