応用プログラム言語II / 演習II #08

ライブラリで楽しよう

情報システム学科
坂本政祐
sakapon@sit.ac.jp

はじめに

- プログラミング言語というのは，たいていの場合，実はそれだけでは大したことはできません。
- C言語なんて，実は純粋な言語仕様だけの状態では，画面への出力すらできません。
  - 「え，printf() あるじゃん」と思うかも知れませんが，あれは実は「標準ライブラリ関数」というものであって，Cの言語仕様そのものでは無いのです。
  - まあ「標準～」というだけあって，どんなCコンパイラにも必ず付いてくるので，区別することに意味があるかはアレですが。

はじめに

- また，この講義の最初の数回でやっていた std::vector や std::map も，あれも実は「STL; Standard Template Library」という，C++に標準で付いてくるライブラリの一部なのです。
- というふうに，ライブラリというのは，広い意味では，「ある言語にそれを追加するといろいろ便利な関数やクラスが使えるようになるソフトウェア部品」ということになります。
  - ちなみに狭い意味: スタティックリンクライブラリ(.lib)やダイナミックリンクライブラリ(.dll)のファイルそのもの。

はじめに

- そこで今回は，ライブラリを活用したプログラミングの手法について学びます。
- これにより，楽に素早くプロトタイプが作れるようになりますよ，という趣旨。

ライブラリとは何か

そのライブラリはどこからきたの

- ライブラリの配付の仕方による分類:
  - 言語に標準でくっついてくるもの（前述の「Cの『標準ライブラリ』」や「C++の『STL』」）
  - 市販のライブラリ
  - オープンソース・ライブラリ，フリーのライブラリ

オープンソース・ライブラリ

- 自分が欲しいような関数はたいてい他人も欲しい。
- そういうのをライブラリ化して公開してくれている団体や個人がいる。
- それを使う側は，同じ機能を自分で再実装しなくても，
  - すばやく，
  - しかも大抵の場合自分で実装するより高品質。
  - 車輪の再発明の防止。

いろんな分野にいろんなライブラリ

- いろんな分野にいろんなライブラリ
  - 画像処理，コンピュータビジョン
    - libjpeg, libpng, libtiff, libmagick, OpenCV, GD
  - 3Dグラフィック
    - OpenGL
  - 拡張現実感
    - ARToolKit
  - 特定のフォーマットのファイルの読み書き
    - libxml2, yaml-cpp,
  - 音声認識
    - Julius

いろんな分野にいろんなライブラリ

applied-progII #08

- （続き）
  - PDF生成
    - libHaru
  - 画面描画汎用，ゲームに使われることが多い
    - DXライブラリ, SDL
  - 圧縮
    - zlib
  - フォントを扱う
    - freetype
  - 物理演算
    - Box2D, Bullet

9

いろんな分野にいろんなライブラリ

applied-progII #08

- （続き）
  - 数値計算
    - BLAS, LAPACK
  - 暗号化
    - OpenSSL

10

実験でOpenGLはやったけど

applied-progII #08

- このうち，OpenGLについてはITコースの人は3年前期の実験でやった。
  - ので，Cに元々用意されている関数以外の，
  - OpenGLに用意されている関数群を使って，
  - でもあくまでCの文法にのっとってコードを書く，というのはなんとなく雰囲気はつかめているのではないかと思います。

11

全部お膳立てされていました

applied-progII #08

- ただし，OpenGLについては，情報基盤センターの課員さんに頼んで，すべてのPCに予めインストールしてもらってあった。
  - OpenGLのセットアップから始めましょう！とかやっていたらたぶん大混乱になるからね。
  - ので，Visual Studioを開いたらすでに使える状態に整っていた。
  - その意味では，本当にそれが元々C言語にない機能なのか，という実感や，ライブラリプログラミングをしている，という実感は少し薄かったんじゃないかと思います。
  - また，「何かのライブラリを Visual Studio から『見える』状態にするには何をどう設定するのか」というのも未経験だと思います。

12

applied-progII #08

- というわけで今どきは:
  - ライブラリを使ったプログラミングなんて，ごく普通。
  - 車輪の再発明なんて，無意味。
    - 学習自体に目的がある場合を除く。
  - 自作ソフトウェアの開発期間を大幅に短縮できる。
  - 開発環境が整ってきたおかげで: 他作ライブラリを自作プログラムから使うのが昔よりも楽。
    - コーディング自体が楽
    - 関数仕様，クラス仕様の参照が楽

13

applied-progII #08

- というわけで今どきは（続き）:
  - ネット環境が整ってきたおかげで: 欲しいライブラリはGoogle先生に聞けば，（在れば）すぐみつかる。
    - 昔は掲示板や NetNews で聞いたりしてました

14

applied-progII #08

- ただしライセンスには注意。
  - それぞれのライブラリには，作者がいて，その作者が決めた「ライセンス」がある。
  - 「オープンソースでフリーなら，どうにでも自由に使っていい」わけではない。
  - 再配布して良いのか，そのライブラリを自作プログラムに組み込んだ場合ソース公開義務があるか・権利表示義務はあるか…
  - オープンソースライブラリで良く用いられるライセンス:
    - GPLv2, GPLv3, LGPL, BSD License, New BSD License, Apache Software License, Creative Commons
    - それぞれ細かく条件が異なる。
    - もちろん，これ以外に作者が自由に決めても良い。

15

applied-progII #08

- 2007年にSCEが発売したPS2のゲーム「ICO」がGPLに違反していると話題になった。
  - ICOが使っていたらしいのは libarc という「アーカイブファイルを統一的に扱える」ライブラリ。有志が ICO のバイナリを解析して発覚（ちなみにSCEがバイナリ解析を禁じていたかどうかは不明）。
  - libarc のライセンスはGPLv2。GPLv2では，それを使用して作成した派生物に関しても，ソースコードの公開義務がある。libarc の作者が「このライブラリはGPLv2です」と決めた以上，それを承知で使用したSCEには，ICOのソースコードを公開する義務がある。
- 将来的に公開・販売を考えているソフトウェアを作成するときは，どんなに便利そうなライブラリがあったとしても涙を飲んで再実装する必要がある。

16

applied-progII #08

- そこで，今回は:
  - ライブラリを自分で配布元からダウンロードしてきて，
  - 自分のプロジェクトからそれを参照できるように Visual Studio 上で設定をして，
  - 実際に自分のコードでそのライブラリの機能を使う，
  - ということをやりたいと思います。

17

今日の狙い

applied-progII #08

- ねらい:
  - 世の中にはライブラリというものがあることを知ってもらうこと。
  - ライブラリを使えば，自作プログラムの開発期間を短縮できることがあることを知ってもらうこと。
  - ライブラリプログラミングをするためには，ダウンロードしたライブラリに対して何から始めれば良いかのコツを知ってもらうこと。
  - Visual Studio で他作ライブラリを使うためにはどこにどのような設定をすれば良いかを知ってもらうこと。
  - どんなライブラリもすべて無条件で使えるわけじゃないことを知ってもらうこと（技術者は技術さえ詳しければいいというわけにもいかないこと）。

18

ライブラリの内容物

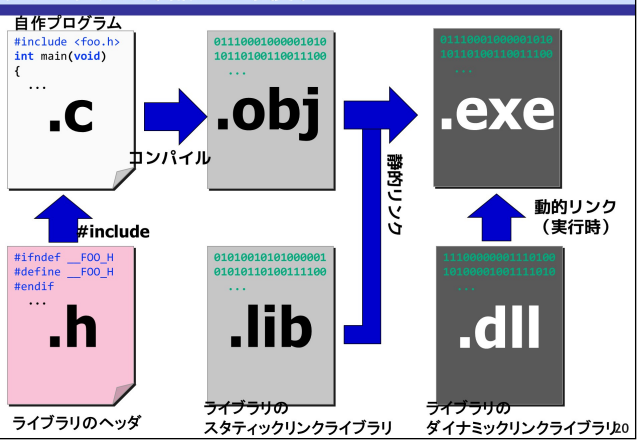
applied-progII #08

- ライブラリに普通含まれるもの:
  - ヘッダ (.h)
    - プログラマが自作するソースから #include で取り込んで使う。
  - スタティックリンクライブラリ (.lib)
    - プログラマが自作したソースと静的リンクされることを想定。
  - ダイナミックリンクライブラリ (.dll)
    - 実行ファイル(.exe)を実行するときに動的リンクされることを想定。

19

ライブラリの部品はどう使われていくか

applied-progII #08



20

applied-progII #08

- ライブラリの配付形態の分類
  - ヘッダとバイナリでの配付
    - 関数のプロトタイプ宣言だけが記述された .h と，コンパイル済みの .lib や .dll が zip などにまとめて配付されている。
    - この形態が多い。
  - 未コンパイルソースでの配付
    - ライブラリ自体のソースがコンパイルされていない状態で配付される。
    - 使いたい人は自分でコンパイルしてから使う。
    - 今日やる libjpeg はこの形式。
  - ソースでの配付
    - ユーザのライブラリと同時にコンパイルされることを想定しての配付。
    - クラス定義が書いてあるヘッダだけを配付するというのがC++用ライブラリでよくある。

21

applied-progII #08

- ライブラリを使うための一般的な手順。
  - ライブラリをダウンロードする。
  - その中に同梱されているであろうドキュメント（大抵の場合，README というファイル）を熟読する。
    - バイナリ配付なのか，ライブラリ自体をコンパイルする必要があるのか。
    - ライセンスの確認。
    - APIの勉強。どんな変数，定数，関数，クラスが用意されていて，それらはどういう仕様なのか。
  - 自作プロジェクトに組み込んでビルド。
  - フリーソフトや有料ソフトとして配付する場合は，使用したライブラリのライセンスを順守して配付。

22

実際のライブラリを使ってみよう

23

applied-progII #08

- 使うライブラリは「libjpeg」。
  - オープンソース。
  - ライセンスは，有名どころのライセンスではなく，独自のフリーライセンス（自由に使用可。自由に再配布可）。商用利用化。ただし libjpeg を使用していることを表示する必要あり。
  - 配付元: http://www.ijg.org/
  - 何ができるライブラリか:
    - jpegファイルの読み込み（デコード）。
    - jpegファイルの書き出し（エンコード）。
    - jpegファイルの中身の操作。
    - つまり，今まで「デジカメが勝手に作る形式」とか「Photoshopとかでしか加工や保存ができない」と思っていたjpegファイルが，実はプログラムから操作できてしまう!

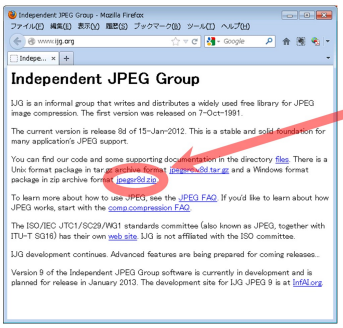
24

applied-progII #08

- libjpeg の配布形態:
  - 未コンパイルソースでの配付。
  - 自分でコンパイルして .lib を作る。
  - その .lib と，あといくつかの .h を自分のプロジェクトから使う。
  - .dll は無し。

libjpegのダウンロード

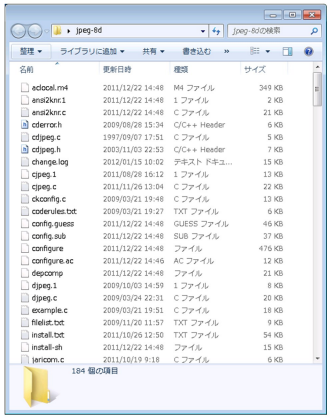
applied-progII #08



- http://www.ijg.org/ に行く。
- jpegsr8d.zip をダウンロード。
- jpegsr8d.zip を伸張し，現れた jpeg-8d というフォルダを C:¥cygwin¥home¥free に移動。

libjpeg の配付物一覧

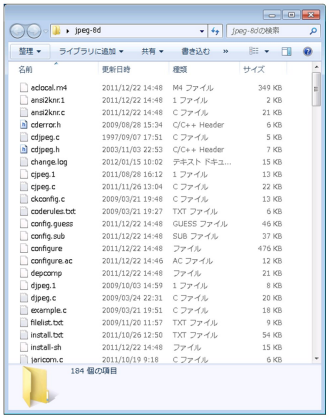
applied-progII #08



- jpegsr8d.zip を伸張すると jpeg-8d というフォルダが現れる。
- このフォルダには 184個もファイルが入っている。これが，libjpeg 自身のソース。これをこれからまずコンパイルして .lib を作る。
- jpeg-8d フォルダを C:¥cygwin¥home¥freeに移動。

libjpeg の配付物一覧

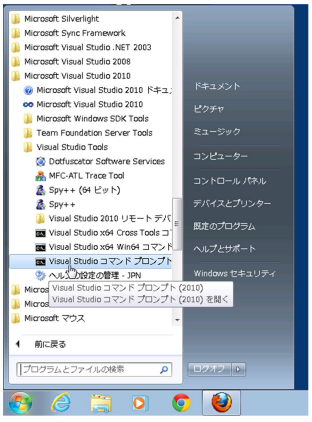
applied-progII #08



- そのためにこのフォルダ内の README というファイルをまず開いて読む。
  - 拡張子なんて甘っちょろい配慮は無いので，ダブルクリックしても開けない。ハッカーの世界は色々厳しい。
- 読めたらjpeg-8d フォルダを C:¥cygwin¥home¥freeに移動。

Visual Studio コマンドプロンプト

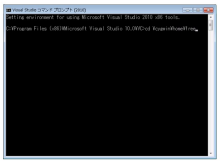
applied-progII #08



- スタートメニューから「Visual Studio コマンドプロンプト (2010)」を開く。
  - これは，コマンドプロンプト上で Visual Studio を CUI で使うための環境。

Visual Studio コマンドプロンプト

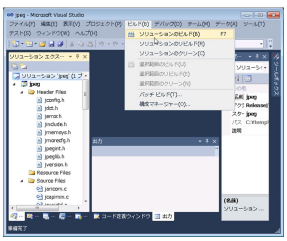
applied-progII #08



- VSコマンドプロンプトで以下のように打つ。これは，libjpeg 自体をビルドするための準備。
  - `cd ¥cygwin¥home¥free¥jped-8d`
  - `nmake /f makefile.vc setup-v10`
- jpeg-8d フォルダに「jpeg.sln」というファイルができたことを確認する。
- VSコマンドプロンプトを閉じる。

libjpeg のビルド

applied-progII #08



- さきほど確認した jpeg.sln をダブルクリック。
- Visual Studio が立ち上がる。
- ビルド→ソリューションのビルド。
- jpeg-8d フォルダの中に jpeg.lib というファイルができたのを確認。
- この .lib と，あと jpeglib.h, jconfig.h, jmorecfg.h を自分のプロジェクトから使えばよい。

自作プロジェクトへ取り込んでみる

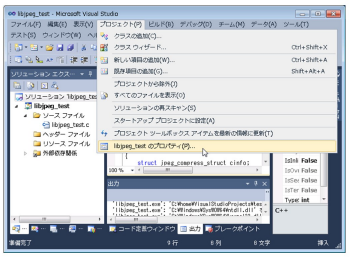
applied-progII #08

- Visual Studio 2010 を立ち上げ，いつものようにWin32コンソールアプリケーションで，空のプロジェクトを作る。
- エクスプローラで，そのプロジェクトの .vcxproj ファイルがあるフォルダを開く。そこに以下の2つのフォルダを作る。
  - include
  - lib
- include フォルダに，jpeg-8d フォルダから以下をコピー。
  - jpeglib.h, jconfig.h, jmorecfg.h
- lib フォルダに，jpeg-8d フォルダから以下をコピー。
  - jpeg.lib

自作プロジェクトへ取り込んでみる

applied-progII #08

- このプロジェクト上で1つソースを作る。
  - 例えば別紙の libjpeg_test.c のような。
- 先ほどコピーしたファイルをVSに認識させるためにプロジェクトのプロパティを開いて，以下のことをやる。

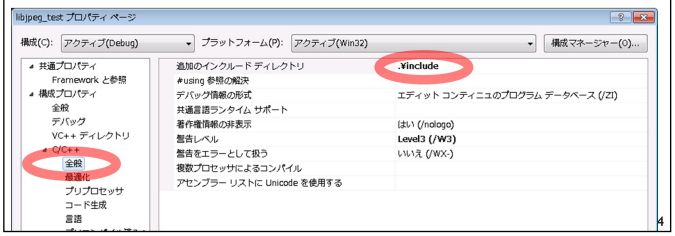


33

.h を見つけられるようにするための設定

applied-progII #08

- 構成プロパティ → C/C++ → 全般
  - 追加のインクルードディレクトリ: `.¥include`
  - これは，.vcxproj から見た相対パスで指定する。
  - 「.」というのはカレントディレクトリなので，「.¥include」なら，「.vcxproj から見てカレント（同じ）ディレクトリのその下の includeというディレクトリ」という意味になる。



4

.lib を見つけられるようにするための設定

applied-progII #08

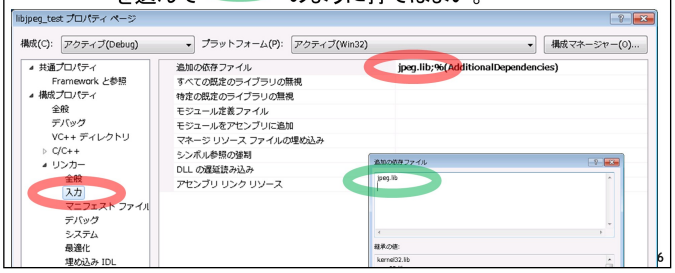
- 構成プロパティ → リンカー → 全般
  - 追加のライブラリディレクトリ: `.¥lib`
  - これも，.vcxproj から見た相対パスで指定する。

35

スタティックリンクしたいファイルの指定

applied-progII #08

- 構成プロパティ → リンカー → 入力
  - 追加の依存ファイル: jpeg.lib を追加。
  - これはファイル名のみ。
  - 追加の仕方: 追加の依存ファイルの右端の ▼ から編集... を選んで のように打てばよい。



6

applied-progII #08

- 以上の設定でビルドできる。
- libjpeg 以外の他のライブラリでも大体似たようなもの。
  - 未コンパイルソース配付の場合はスライド30から，
  - バイナリ配付ならスライド31からと同様の手順。
  - .dll がある場合は，実行ファイルと同じフォルダに置く。
  - こうすることで，「ポータブル」なプロジェクトになる。

37

applied-progII #08

- という感じで，自分で同じ機能を実装するのに比べれば，この程度の設定は慣れれば全然大変じゃないので，積極的にライブラリを使って楽しましょう。というのが今日の結論。

38

今日の課題

applied-progII #08

- 課題1: libjpeg に関して，以下に取り組んで下さい。
  - libjpeg 自体をビルドして下さい。
  - libjpeg を自作プロジェクトに取り込んでください。
  - 別紙の libjpeg_test.c を写経してビルドし，問題なくビルドできることを確認してください。
  - 実行してみて（一瞬で終わりますが，それで正常です。ファイルを生成するプログラムですから），.vcxproj と同じフォルダに libjpeg_test.jpg ファイルが生成されることを確認してください。

39

今日の課題

applied-progII #08

- 課題2: せっかくなので，libjpeg_test.c を改造して，何か幾何学的模様や何か図形が表示されるようにしてください。
  - 課題1と課題2は連続してやって，提出するのは1つで良いです。そのプロジェクトのフォルダを zip 圧縮して提出して下さい。
  - つまり今日提出してもらうのは zip ファイルひとつのみです。

40

今週の落穂拾い

applied-progII #08

- 90年代とかに比べれば，今は良質のライブラリがずらりと揃っていて羨ましいです。
- その理由としては，ネット環境が良くなった，sourceforge とか GitHub とかの OSS ホスティングサービスが発達した，統合開発環境も頭が良くなった，OOPLなら殊更モジュール化しやすい，等色々あると思います。
- ただ，GPLが過激なのでGPL汚染なんて言葉までできちゃったのはちょっと悲しいことです。
- お米を食べるときは農家の方に感謝するのと同様，ライブラリを使うときはその作者の方に感謝しながら使いましょう。ありがたや。

41