

コンパイラ及び演習

関澤 俊弦

日本大学 工学部 情報工学科

- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- プログラミング技法
- 演習

講義時間と教室

■ 講義時間, 教室

- 月曜 4, 5限, 61-202

■ 担当

- 関澤 俊弦

■ オフィスアワー

- 金曜 3限, 61号館204研究室

シラバス

■ 教育目標

- コンパイラの基本的な概念及び動作原理を理解する
- 字句解析および構文解析の仕組みを理解し、それぞれの処理プログラムを作成できる
- 簡易言語のコンパイラを作成できる

■ 講義概要

- 字句解析, 構文解析, 中間コード生成などの各フェーズを, 理論および演習の両面から学ぶ.

演習に関して

■ プログラミング言語: C言語

- C99の機能は断りなく使用できるとします
 - ・ コメント(//) , bool型など

■ サンプルコード

- 必要に応じてサンプルコードを提示します
- サンプルコードは、講義の要点を示すもの
 - ・ エラー処理などは、最低限しか実装されていません
 - ・ 各自が読んで理解することを期待しています

■ 採点システム

- 毎週の演習は、採点システムを使用します

備考

- 「オートマトンと言語及び演習」
 - 履修していることが望ましい

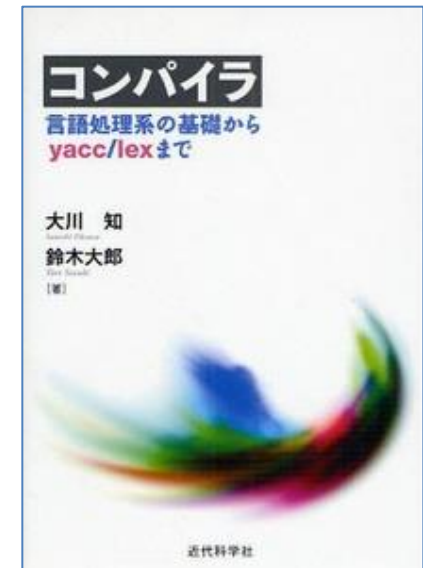
教科書・参考書

■ 教科書

- 大川知, 鈴木太郎著:
コンパイラ – 言語処理系の基礎からyacc/lexまで,
近代科学社

■ 参考書

- 必要に応じて講義中に指示する



授業計画

1. イントロダクション
2. 文, 形式言語, 文法
3. オートマトン
4. BNF記法, 字句解析I
5. 字句解析II
6. 構文解析I
7. 構文解析II
8. 中間試験
9. 構文解析III
10. 意味解析
11. 中間コード生成I
12. 中間コード生成II
13. コンパイラの作成I
14. コンパイラの作成II
15. 期末試験

成績評価

■ 成績評価

演習課題, 中間試験, 期末試験で評価する

- 演習課題: 20%
- 中間試験: 30%
- 期末試験: 50%

- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- プログラミング技法
- 演習

計算機と言語処理系の歴史

1. 初期の計算機

- プログラムはスイッチと配線により回路を構成して実現した

2. プログラム内蔵型計算機の出現

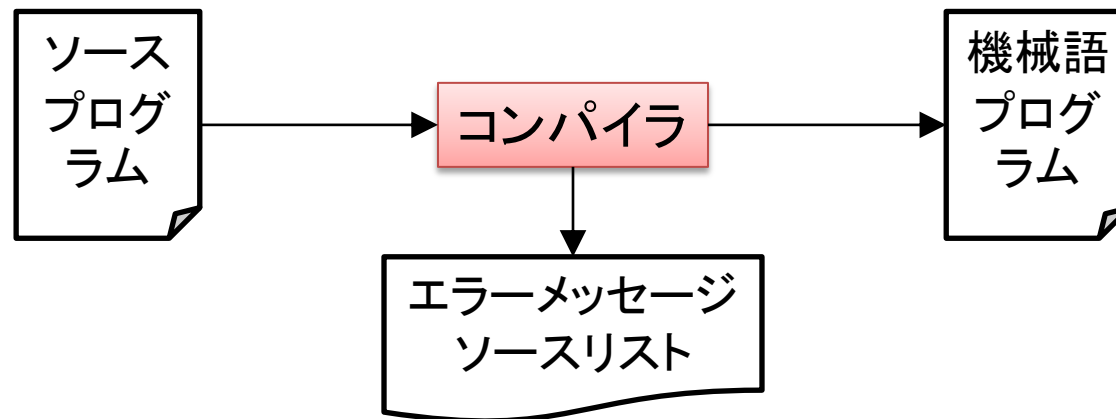
1. 計算機に依存した機械語でのプログラミング
2. 機械に依存しないアセンブリ言語
3. 高級プログラミング言語の登場
 - FORTRAN, COBOL, LISP, ...
 - 機械語に変換する言語処理系を伴う
(変換プログラム)

言語処理系

- プログラムの開発・翻訳・実行に関するプログラム群の総称
 - ある言語で記述されたプログラムを, 他の言語で記述されたプログラムに変換するシステム.
- 各種の言語処理系
 - コンパイラ,
 - インタプリタ,
 - アセンブラ,
 - トランスレータ, ...

言語処理系: コンパイラ

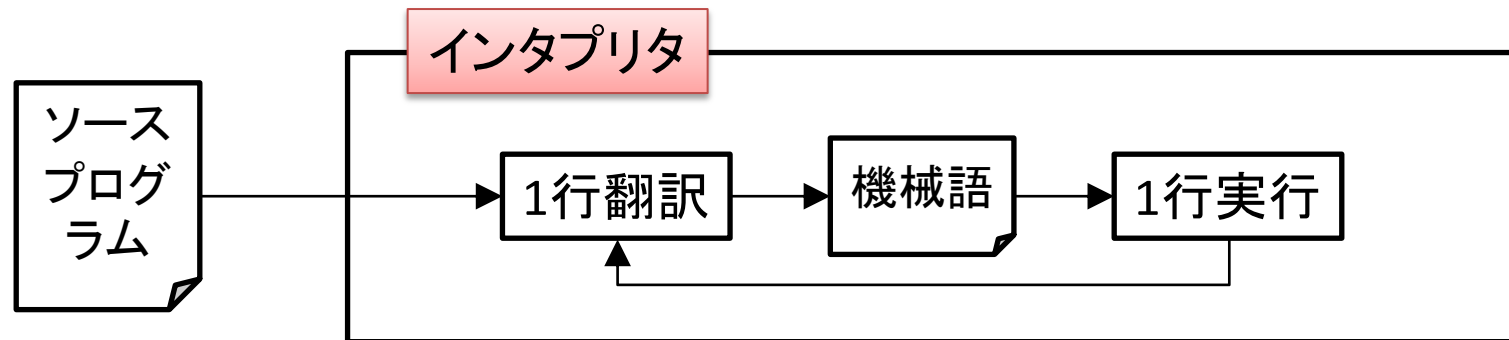
- ソースプログラムを, 機械語プログラムに変換する
 - 一般に, ソースプログラムは高級プログラミング言語で記述される



高級プログラミング言語: 人間にとって理解しやすい言語
低級プログラミング言語: 機械向けの言語

言語処理系: インタプリタ

- ソースプログラムを, 1文ずつ逐次解釈して実行するソフトウェア
 - プログラムの変換は行なわない



	コンパイラ	インタプリタ
出力	機械語	中間語
翻訳実行	長い	短い
実行時間	速い	遅い

言語処理系: アセンブラ, トランスレータ

■ アセンブラ

- 記号言語(アセンブリ言語)で記述されたプログラムを機械語プログラムに変換する
 - 記号言語は, 原則として, 機械語に1対1に対応する

■ トランスレータ

- あるプログラミング言語で書かれたプログラムを, 他のプログラミング言語で書かれたプログラムに変換するプログラム

Tips: その他の方式

■ コンパイラ・インタプリタ

1. コンパイラがソースプログラムを中間コードに変換,
2. 中間コードをインタプリタが逐次実行する方式

■ ジャストインタイム・コンパイラ

□ 中間コードをインタプリタ内でネイティブコードにコンパイルして実行する方式

- Javaで採用されている
- ユーザからはコンパイラ・インタプリタと同等に見える
 - 実行速度が上がることが多い
 - コンパイラ起動時に不自然に時間がかかることがある

- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- プログラミング技法
- 演習

用語

計算機上のプログラムの種類:

■ ソースプログラム (ソースコード)

- 計算機への指示を, 人間が読み書き可能な人工言語 (プログラミング言語) で記述したもの

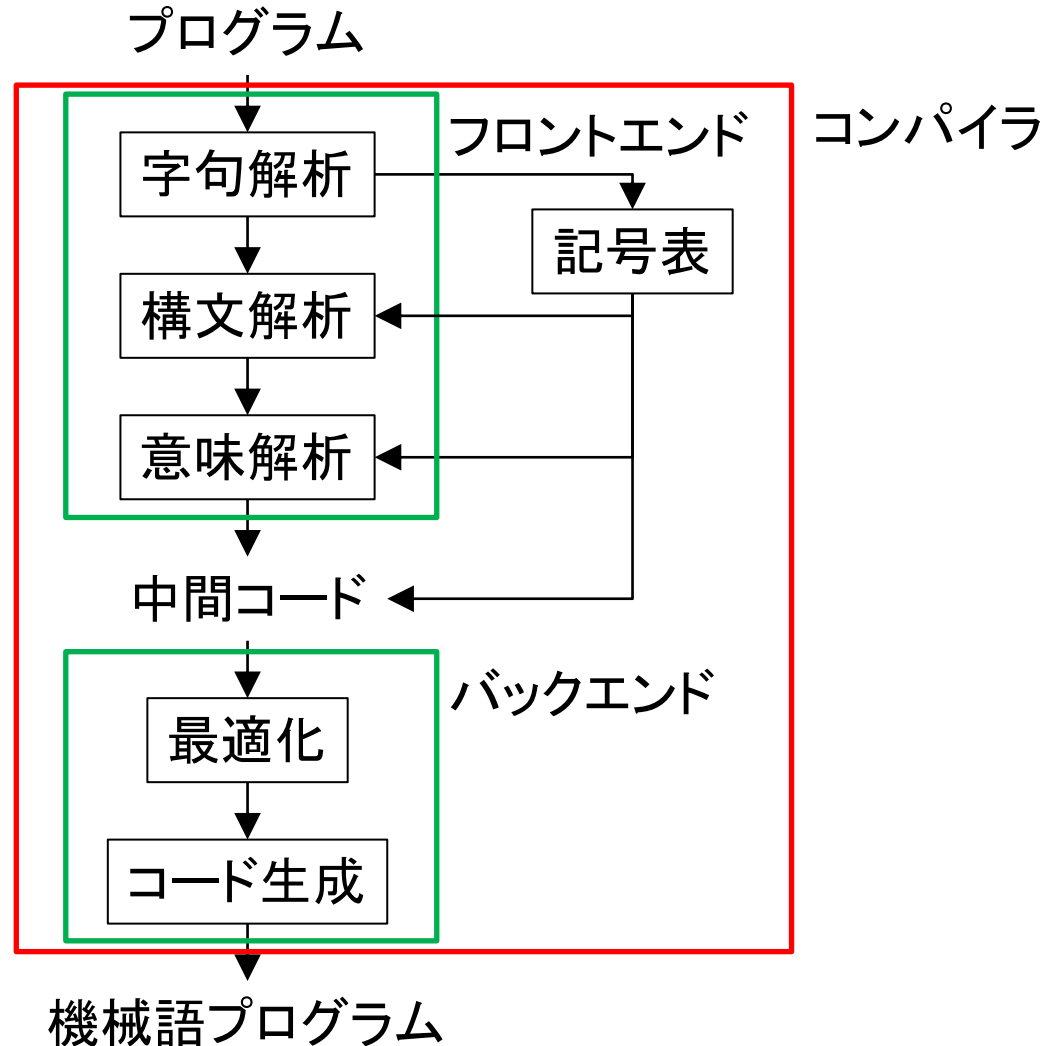
■ 中間コード

- ソースプログラムの解析結果に基づき作成されるプログラム

■ 機械語プログラム

- 機械が実行可能なプログラム
 - “機械コード”, “目的プログラム”とも呼ばれる

コンパイラの処理過程



コンパイラの処理過程(フェーズ)

■ フロントエンド

□ 字句解析

- ・ ソースプログラムに含まれる字句を抽出する

□ 構文解析

- ・ 抽出された字句に基づき, プログラムの構文を解析する

□ 意味解析

- ・ 構文解析に基づき, プログラムの意味を解析する

■ バックエンド

□ 最適化

- ・ 実行時間など, 最適化された等価なコードを生成する

□ コード生成

- ・ 意味解析と構文解析から, 機械語プログラムを生成する

- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- プログラミング技法
- 演習

動作概要: 字句解析

例: 台形の面積を求める次のプログラム

```
area=1/2*(upperbase+lowerbase)*height
```

■ 字句解析

□ ソースプログラムに含まれる字句の抽出

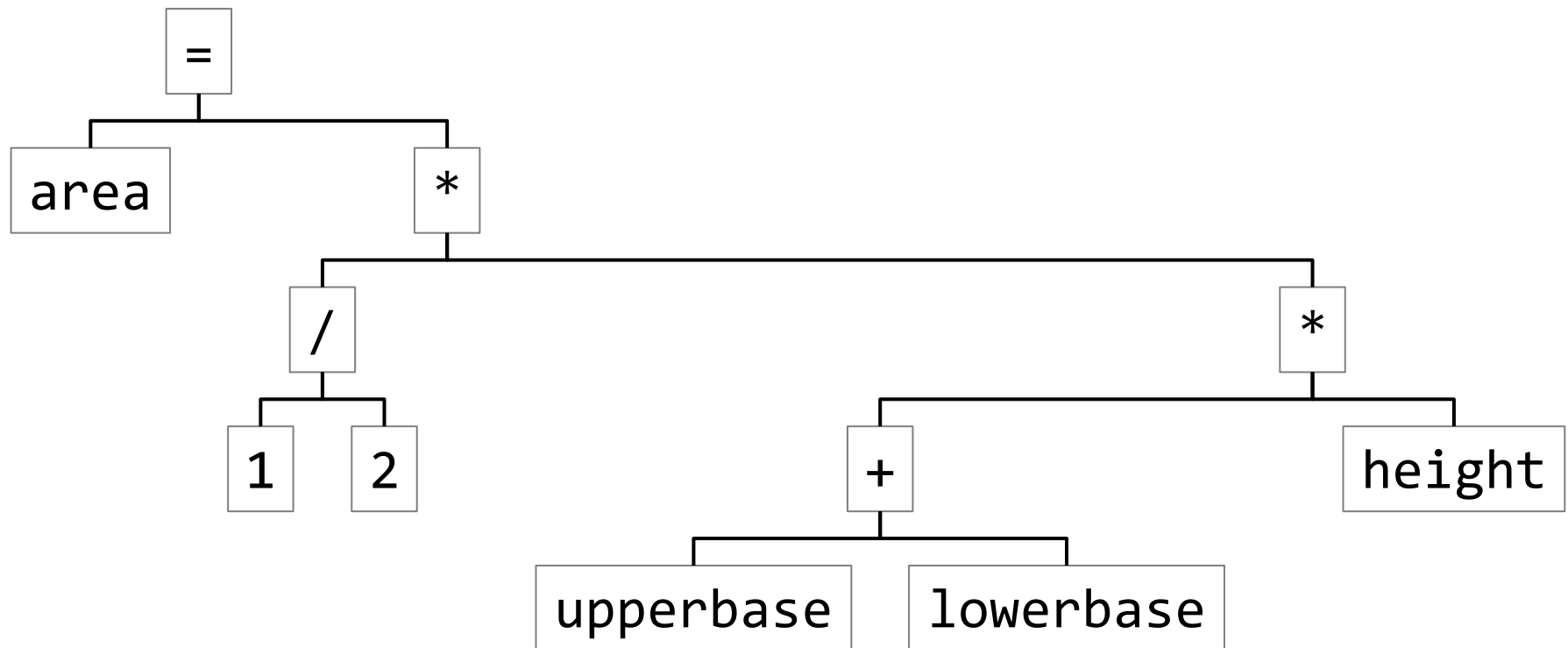
area = 1 / 2 * (upperbase + lowerbase) * height

動作概要: 構文解析

■ 構文解析

□ 抽出された字句に基づく構文の解析

- 例: 構文木の生成

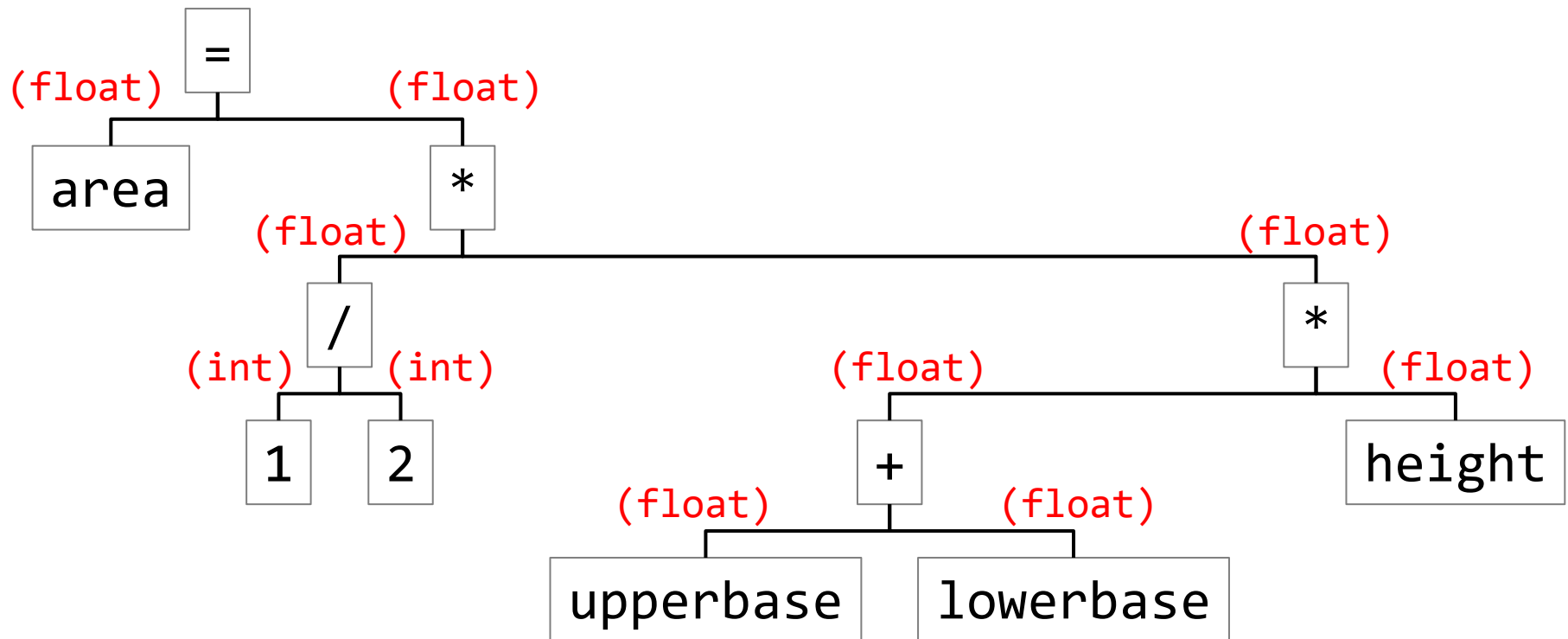


動作概要: 意味解析

■ 意味解析

□ 構文解析に基づくプログラムの意味の解析

- 例: データ型の補完

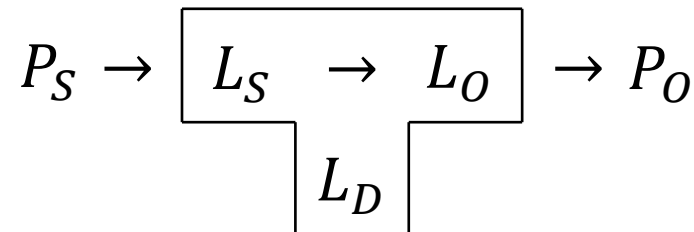


- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- プログラミング技法
- 演習

T図式

■ 言語処理系の概要を図的に表現したもの

- P_S : ソースプログラム
- P_O : オブジェクトプログラム
- L_S : ソースプログラミング言語
- L_O : オブジェクトプログラミング言語
- L_D : 処理系の記述言語

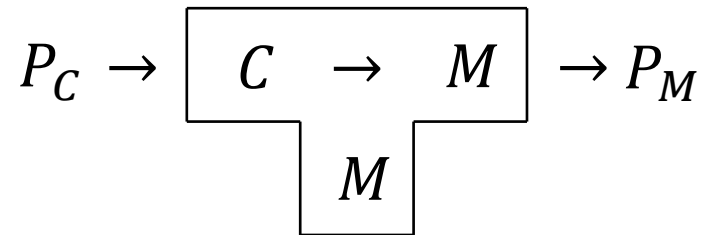


T図式を組合わせ, 新しいT図式を得ることができる

T図式: Cコンパイラ

■ C言語のコンパイラのT図式

- P_C : ソースプログラム
- P_M : オブジェクトプログラム
- C : ソースプログラミング言語
- M : オブジェクトプログラミング言語



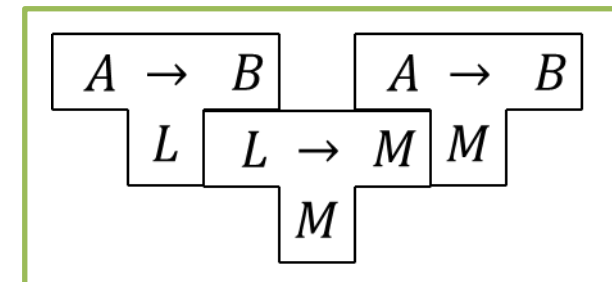
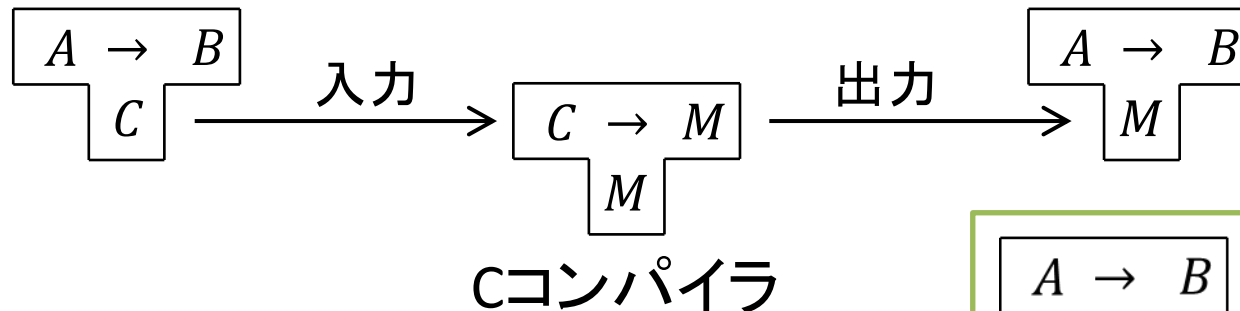
オブジェクトプログラミング言語と処理系の記述言語が、共に M の場合の例

T図式: トランスレータの例

- C言語で記述されたA言語プログラムからB言語プログラムへのトランスレータを, 実行可能な機械語Mに変換する

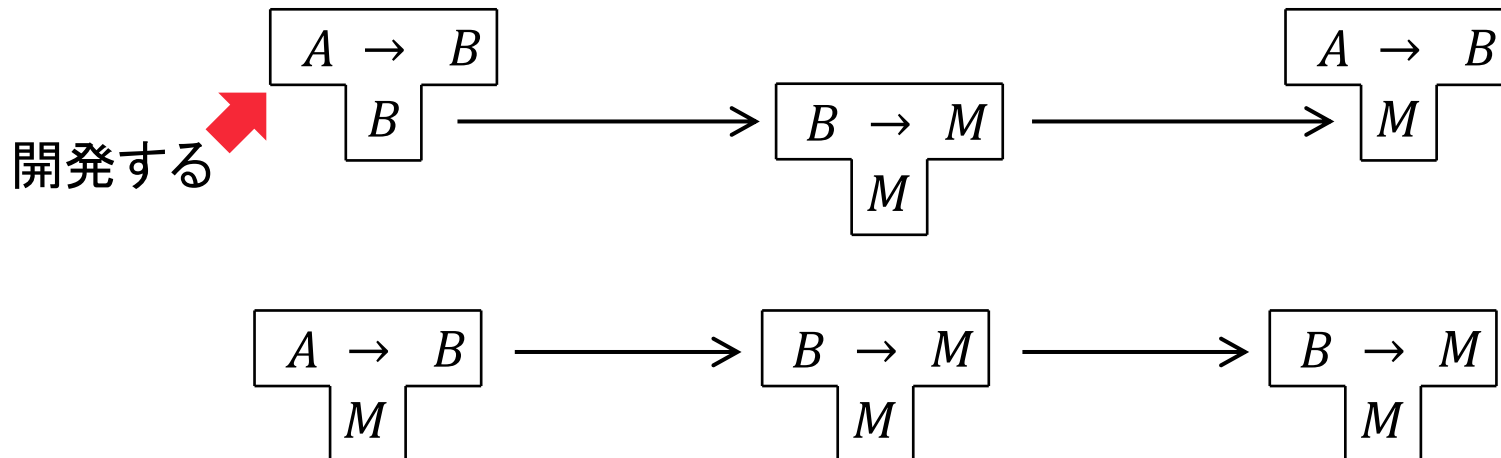
C言語で記述された
AからBへの
トランスレータ

機械語Mで記述された
AからBへの
トランスレータ



T図式: コンパイラのコンパイル

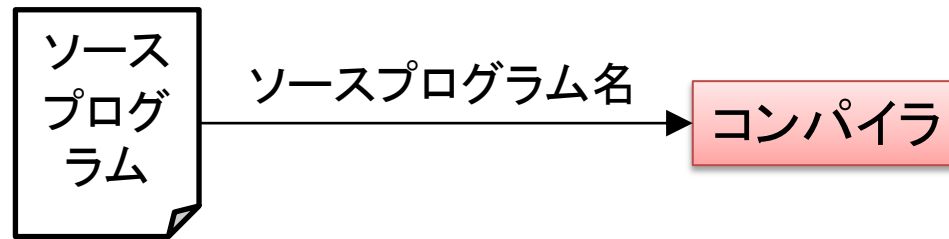
- 機械語 M で記述された A 言語のコンパイラ
 - B 言語のコンパイラは存在するとする
 - 次の組み合わせで作成する:
 - 機械語 M の A 言語から B 言語へのトランスレータ
 - B 言語から機械語 M へのコンパイラ



- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- プログラミング技法
- 演習

コマンドライン引数を扱う

- コマンドラインから, コンパイルするソースコードを指定する
 - コマンドラインの引数を取得する必要がある



gccの例.
コマンドラインで"sample.c"を与える

コマンドライン引数を扱う

■ コマンドライン引数の扱い方

□ main関数の引数で受け取る

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

```
int main(int argc, char *argv[]) {
    ...
    exit(EXIT_SUCCESS);
}
```

int argc (argument count)
引数の個数

char *argv[] (argument vector)
引数で指定される文字列の配列
へのポインタ

■ 例

```
u2X6000@cse-ssh[ ]: ./1-4 ./sample01.c
```

argc = 2
(引数は2個)

argv[0] = "./1-4"
argv[1] = "./sample01.c"

コマンドライン引数を扱う

■ コマンドライン引数の個数と内容を表示するプログラム (example01_4.c)

```
int main(int argc, char *argv[]) {  
    int i;
```

```
    printf("number of arguments: %d¥n", argc);  
    for (i = 0; i < argc; i++) {  
        printf("%d: %s¥n", i, argv[i]);  
    }
```

```
    ...  
}
```

引数の個数 (argc) を表示

引数の文字列 (argv[i]) を
1つずつ表示



この講義では, argv[1]にソースファイルのファイル名を指定し, ファイルを開くことが多い.

- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- 演習

演習

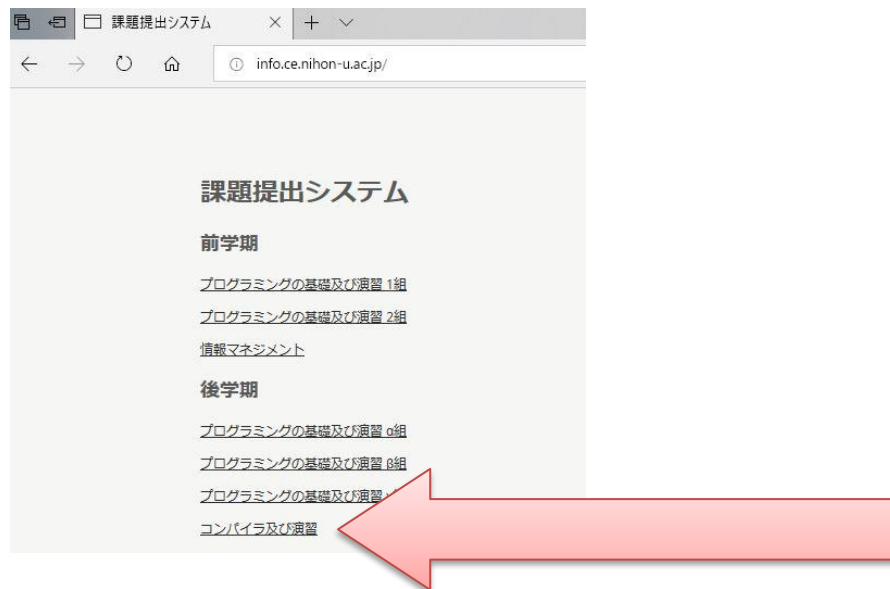
演習1-1: 演習課題提出サイトへの登録

■ 演習課題提出サイトへのユーザ登録

1. ブラウザから, 次のURLにアクセスする.

<http://info.ce.nihon-u.ac.jp>

2. "コンパイラ及び演習"をクリックする.



演習1-1: 演習課題提出サイトへの登録

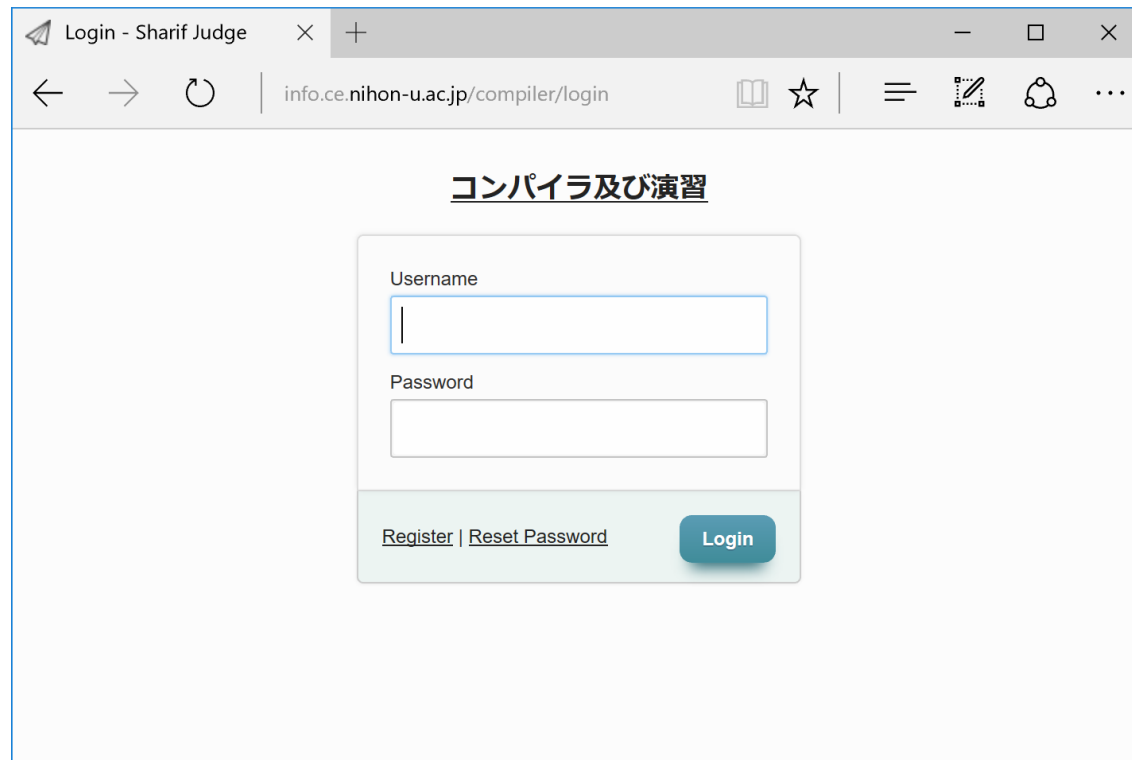
■ 演習課題提出サイトへのユーザ登録手順

1. ブラウザから, 次のURLにアクセスする.
<http://info.ce.nihon-u.ac.jp/compiler/>
2. "Register"をクリックする.
3. "Registration Code", "Username", "Email",
"Password", "Password, Again"を入力し,
"Register"をクリックする.
4. プロファイルの変更

ユーザ登録: 手順1

- ブラウザから, 次のURLにアクセスする.

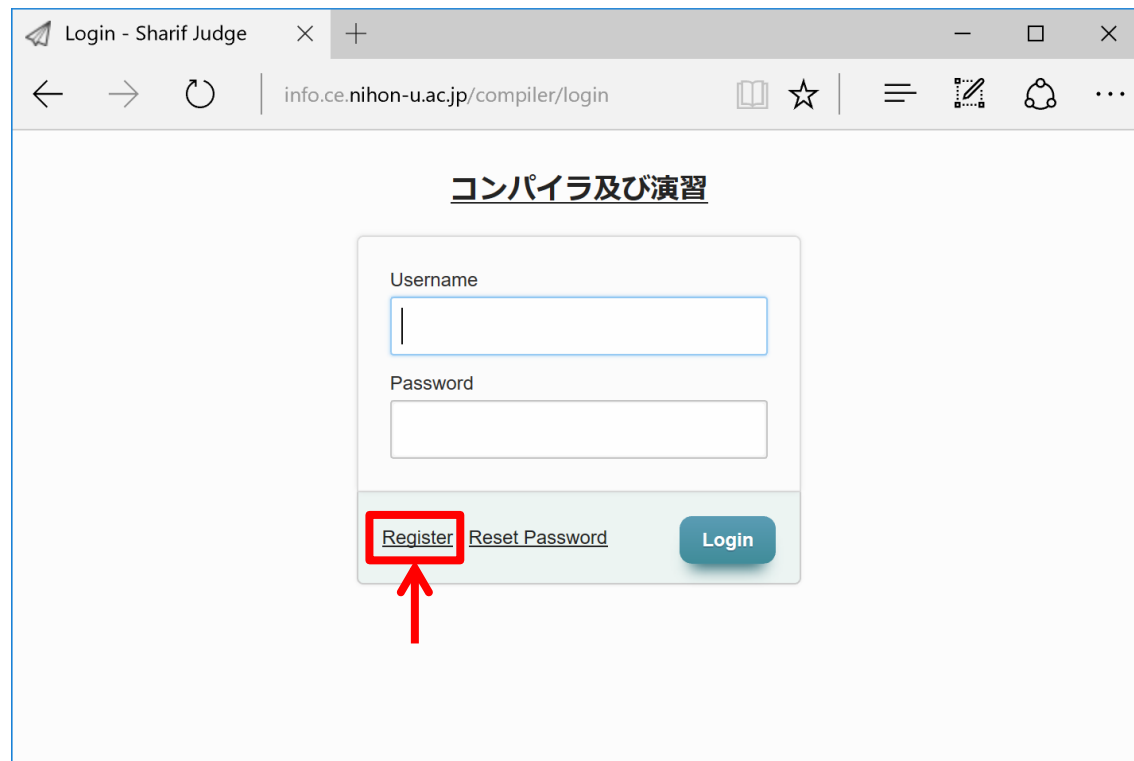
<http://info.ce.nihon-u.ac.jp/compiler/>



The screenshot shows a web browser window with the title "Login - Sharif Judge". The address bar displays the URL "info.ce.nihon-u.ac.jp/compiler/login". The page content features the heading "コンパイラ及び演習" (Compiler and Exercise). Below this, there is a login form with two input fields: "Username" and "Password". At the bottom of the form, there are links for "Register" and "Reset Password", and a blue "Login" button.

ユーザ登録: 手順2

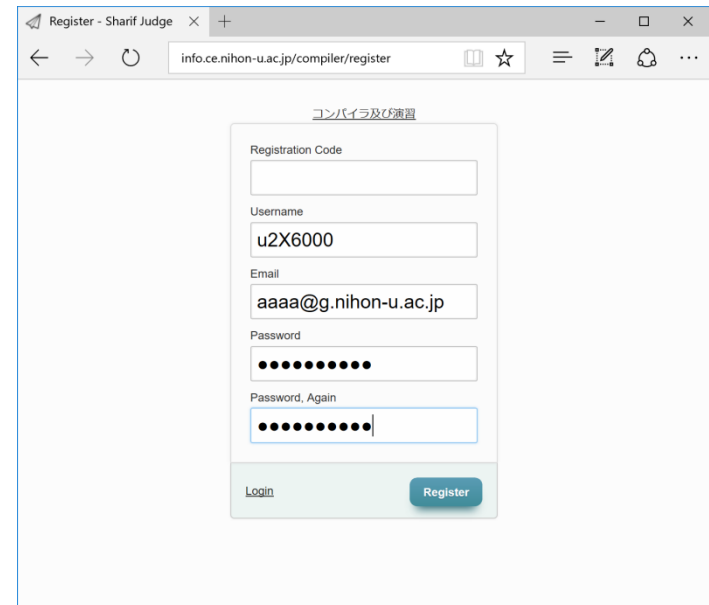
■ "Register"をクリックする



The screenshot shows a web browser window with the title "Login - Sharif Judge" and the URL "info.ce.nihon-u.ac.jp/compiler/login". The page content is titled "コンパイラ及び演習" (Compiler and Exercise). It features a login form with two input fields: "Username" and "Password". Below these fields, there are three buttons: "Register", "Reset Password", and "Login". The "Register" button is highlighted with a red rectangle, and a red arrow points to it from below.

ユーザ登録: 手順3

- 各項目を入力し, "Register"をクリックする
 - "Registration Code": 講義中に掲示
 - "Username": u学籍番号 (iMacと同じ)
 - "Email": 原則として日大の公式アドレス
 - "Password"
 - "Password, Again"



The screenshot shows a web browser window with the title "Register - Sharif Judge" and the URL "info.ce.nihon-u.ac.jp/compiler/register". The page content is titled "コンパイラ及び演習" (Compiler and Exercises). It contains a registration form with the following fields: "Registration Code" (empty), "Username" (filled with "u2X6000"), "Email" (filled with "aaaa@g.nihon-u.ac.jp"), "Password" (filled with dots), and "Password, Again" (filled with dots). At the bottom of the form are two buttons: "Login" and "Register".

ログイン

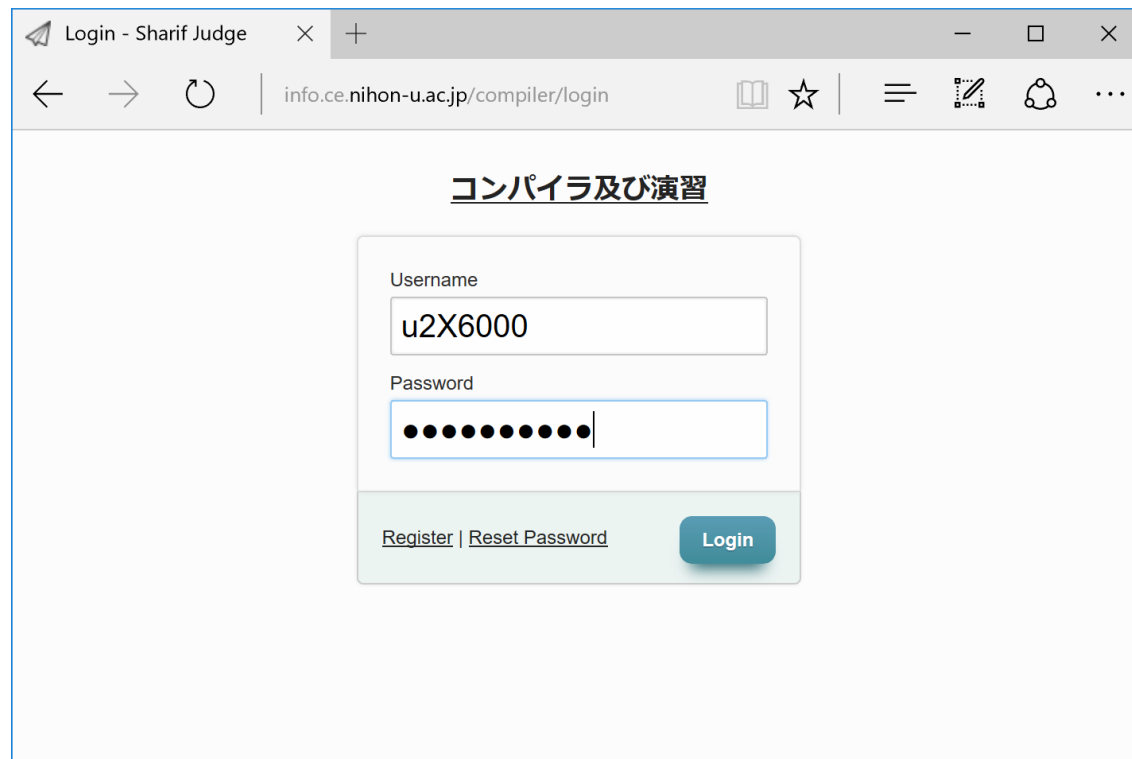
- <http://info.ce.nihon-u.ac.jp/>
 - 「コンパイラ及び演習」をクリック



ログイン

■ username, passwordを入力

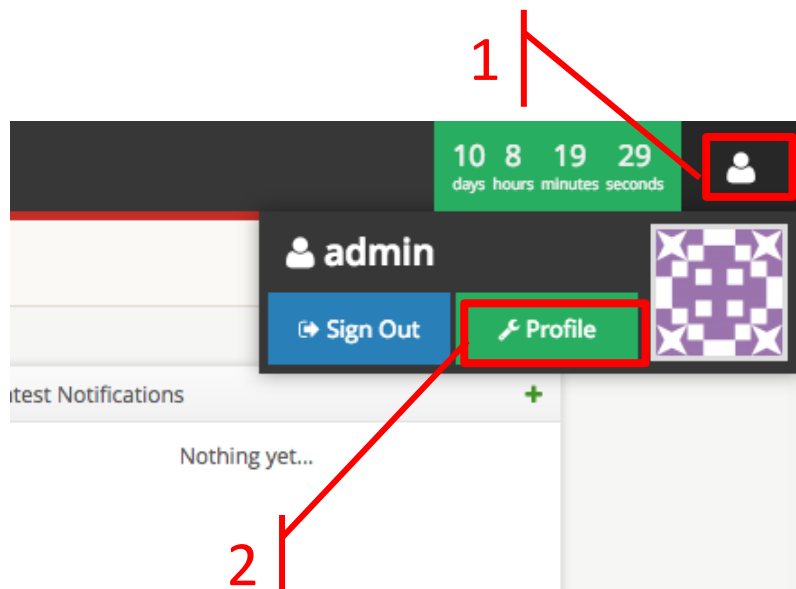
□ <http://info.ce.nihon-u.ac.jp/compiler/>



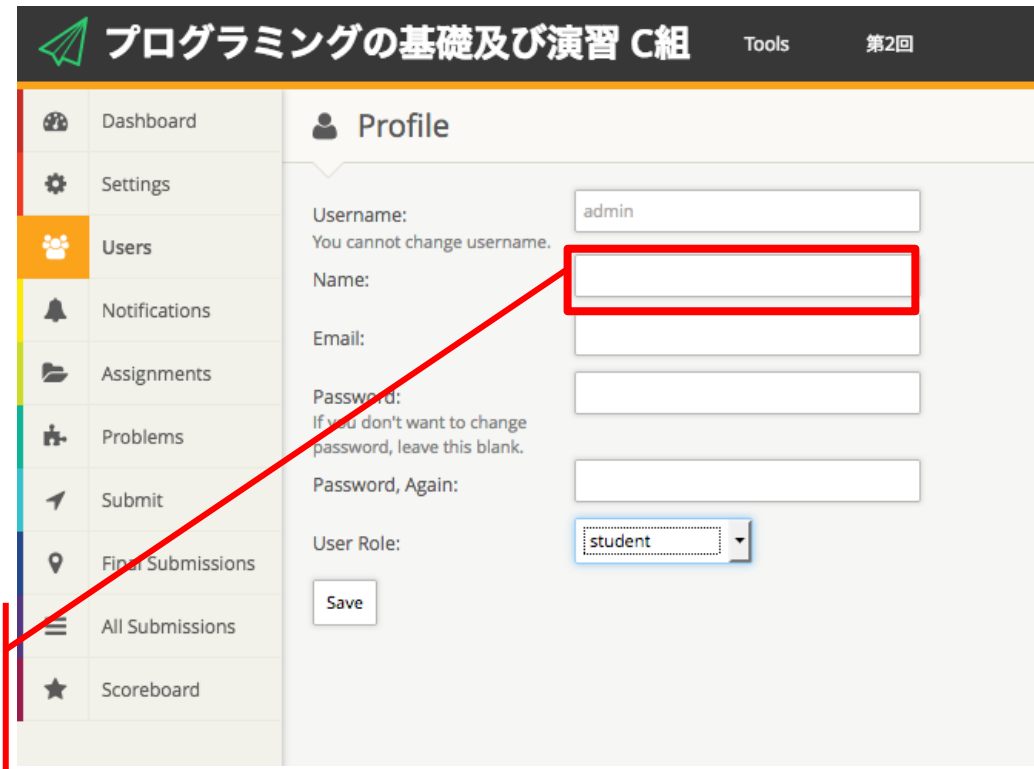
The screenshot shows a web browser window with the title "Login - Sharif Judge". The address bar displays "info.ce.nihon-u.ac.jp/compiler/login". The page content is titled "コンパイラ及び演習" (Compiler and Exercise). It features a login form with two input fields: "Username" containing "u2X6000" and "Password" filled with dots. Below the password field are links for "Register" and "Reset Password", and a blue "Login" button.

プロフィールの変更

1. 右上の人型アイコンをクリック
2. "Profile"をクリック
3. 氏名を記述



氏名を記入
※ 漢字で, ○○ ○○

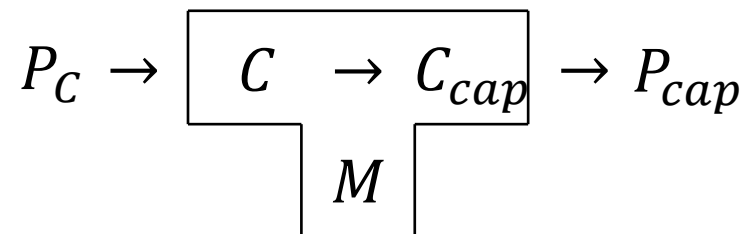


演習1-2: 本講義用ディレクトリの作成

- 各自の計算機環境に、本講義用のディレクトリを作成しなさい
 - ディレクトリ名の制限はありませんが、紛らわしくない名前を付けること
 - 本講義では、相当数のファイルの作成が予想されます

トランスレータccaptの実装

- C言語から仮想言語Ccapitalへのトランスレータ ccapt (CCAPital Transrator) を実装する
 - P_C : C言語のプログラム
 - P_{cap} : Ccapital言語のプログラム
 - C : C言語
 - C_{cap} : Ccapital言語
 - M : 処理系の記述言語



仮想言語Ccapital

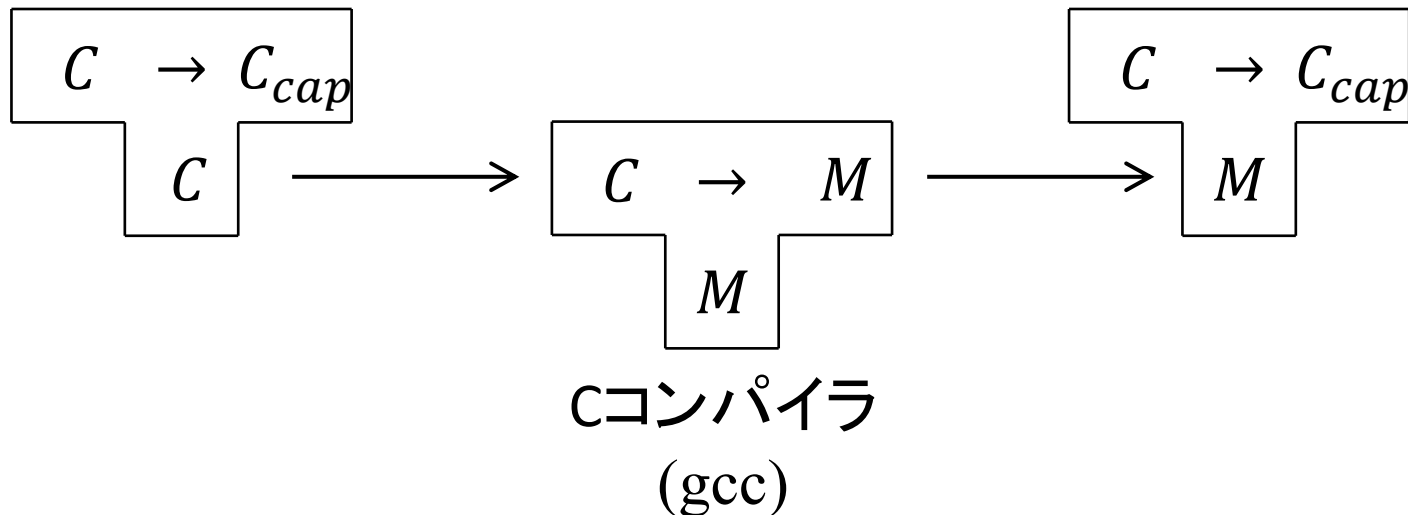
■ 言語仕様

- C言語と同じ文法で定義される. ただし, プログラムはすべて大文字で記述される

トランスレータccaptの実装

■ 手順

1. C言語で, CからCcapitalへのトランスレータを作成する
2. Cコンパイラ(gcc)を用いて, CからCcapitalへのトランスレータをコンパイルする



演習1-3: ツール名表示

- C言語で, 標準出力に
"ccapt: a C-to-Ccapital translator"
と表示するプログラムを作成せよ
 - 文字はすべて半角とする
 - スペースを含めて次の例と同じ出力をすること
- 演習課題サイトで提出せよ(以降, 同様)
 - 点数(Score)が100となることが望ましい

演習1-3: ツール名表示

■ 動作例



```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh[57]: ./1-3
ccapt: a C-to-Ccapital translator
sekizawa@cse-ssh[58]:
```

演習1-4: ファイルの内容表示

- 次の機能をもつプログラムを作成せよ
 - コマンドラインからファイル名を指定して起動する
 - そのファイルが存在する場合, ファイルの内容を標準出力に表示する.
- 条件 (教科書 p.16, 演習問題1.4改)
 - C言語で実装すること
 - ファイル内の1行の最大文字数は256文字とする
(行末記号を含む)



p. 16, 演習問題1.4改

演習1-4:ファイルの内容表示

■ 動作例

- "sample01.c"を入力ファイルとした場合



The screenshot shows a PuTTY terminal window titled "gw.cse.ce.nihon-u.ac.jp - PuTTY". The terminal displays the execution of a C program named "sample01.c". The program's source code is shown as follows:

```
sekizawa@cse-ssh2[36]: ./1-4 ./sample01.c
#include <stdio.h>

int main(void) {
    double base = 10, height = 10;
    double area;

    area = 1 / 2 * base * height;
    printf("area = %lf\n", area);

    return 0;
}
```

The terminal output shows the program running successfully, with a green cursor at the prompt "sekizawa@cse-ssh2[37]:". A semi-transparent tooltip with the text "ウィンドウの種別切り取り(W)" is visible over the code.


演習1-5: トランスレータccapt

- 次の機能をもつプログラムを作成せよ
 - コマンドラインからファイル名を指定して起動する
 - そのファイルについて, 文字が英字の小文字であれば対応する大文字に変換して表示し, 小文字以外の文字の場合そのまま表示する

演習1-5:トランスレータccapt

■ 動作例

- "sample01.c"を入力ファイルとした場合



```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh2[38]: ./1-5 ./sample01.c
#include <stdio.h>

int main(void) {
    double base = 10, height = 10;
    double area;

    area = 1 / 2 * base * height;
    printf("AREA = %f\n", area);

    return 0;
}
sekizawa@cse-ssh2[39]:
```

演習1-5: 参考

■ islower

関数名	int islower(int c)	
ヘッダー	ctype.h	
引数	int c	判定する文字
戻り値	int	cが小文字('a' - 'z')かどうか判定する. 小文字の場合0以外, 小文字以外の場合 0を返す.
機能	引数cが小文字であるか否かを判定する.	

演習1-5: 参考

■ toupper

関数名	int toupper(int c)	
ヘッダー	ctype.h	
引数	int c	変換する文字
戻り値	int	引数cが小文字('a' - 'z')の場合, 対応する大文字の値を返す. 小文字以外の場合, 変換せずにcを返す.
機能	小文字を大文字に変換する.	

演習1-xの提出に関して

- 提出期限: 2018年10月7日(日) 0:00まで
 - 第1回は履修者が確定していないため

まとめ

- 講義に関して
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式
- 演習
 - トランスレータccaptの実装