

# プログラムとは何か

情報処理演習

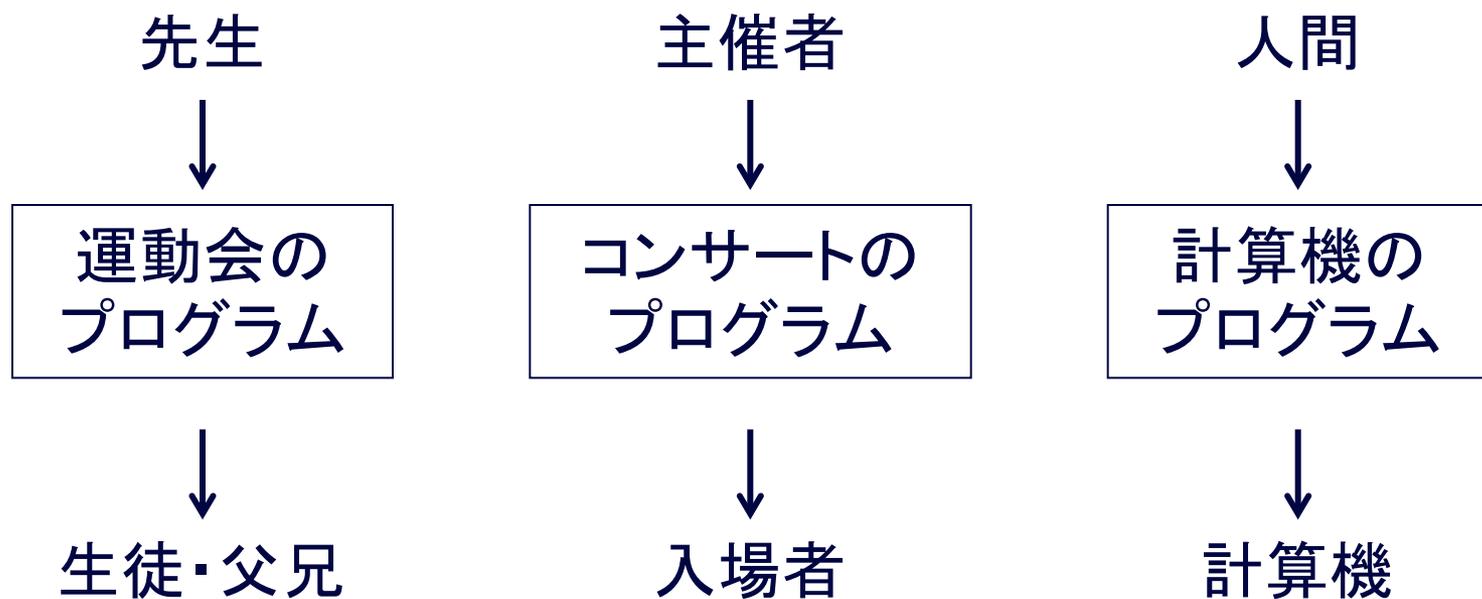
(テキスト: 第1章, 第2章)

# 今日の内容

1. プログラム(program)
2. プログラミング言語
3. プログラミング言語 C
4. なぜプログラミングを学ぶのか？
5. Xcodeが自動生成するC言語のプログラム
6. Xcode統合環境の使い方
7. 今日の練習問題

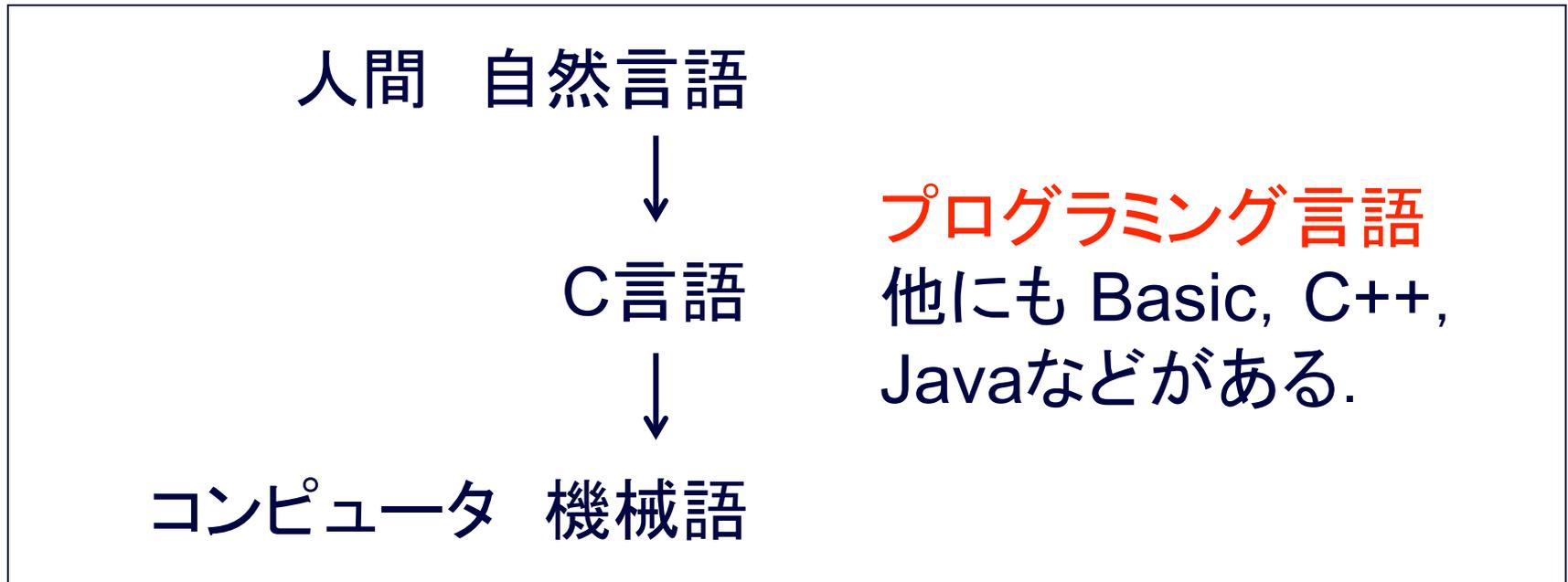
# 1. プログラム(program)

- どのようなことを行うのかを前もって書き記したもの
- 仕事の手順を指示する命令書



## 2. プログラミング言語

### プログラムを書くための言葉



#### プログラミング言語の分類

- 手続き型言語と関数型言語
- コンパイラとインタプリタ

# 手続き型言語と関数型言語

## ■ 手続き型言語

- 処理の手順を記述する言語
- 変数の値を書き換えながら実行が進む.
- C言語, Java言語, Pascal, FORTRANなど

## ■ 関数型言語

- 計算式を定義する言語
- 定義された式を計算することで実行が進む.
- 変数の値を書き換えるような副作用はない(少ない).
- LISP言語, Scheme言語など

# コンパイラ方式とインタプリタ方式

記述したプログラムをどのようにして実行するかで分類

## ■ コンパイラ方式

- コンパイラは、ある言語を別の言語に変換するプログラム
- ソースコードを変換して実行可能コードを生成する.
- 実行可能コードを実行する.
- 一般に、実行速度は速い.

## ■ インタプリタ方式

- インタプリタは、ソースコードを逐次解析しながら実行するプログラム
- 実行可能コードは生成しない.
- 一般に、実行速度は遅い.

### 3. プログラミング言語C

- D. M. Ritchieが1970年にUNIXを記述するために作成した言語
- ハードウェアを直接扱う操作も記述できる.
- ANSIにより規格化されており, 汎用性が高い.
- 手続き型言語のひとつ
- コンパイラ方式の言語のひとつ

# 計算機プログラムとは？

## ■ 計算機プログラムとは？

- ある問題を解くための**計算の手順(順序)書**であり、  
その問題の**解法の定義書**である。

2000円の予算で、5人にアイスクリームを買う。

## ■ 計算の要素

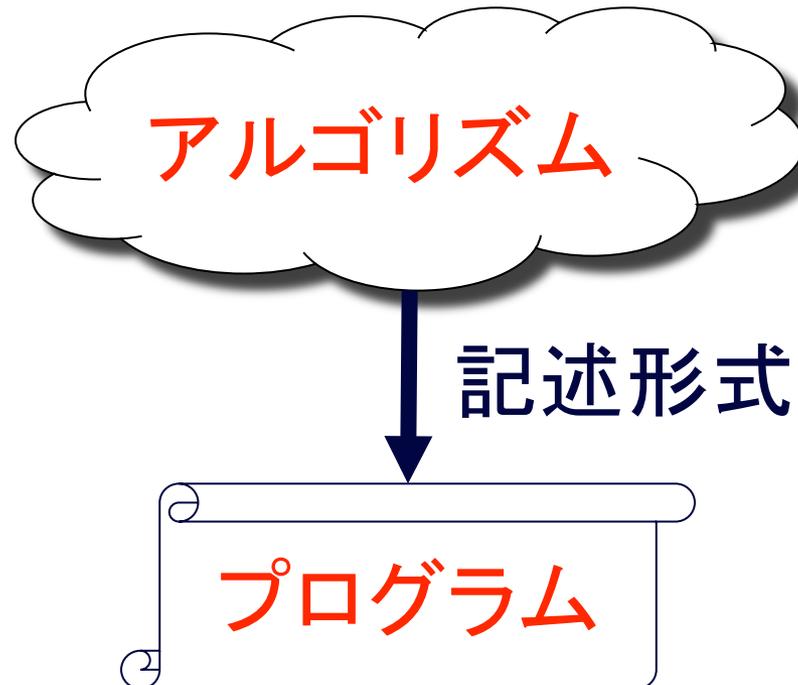
- 数値演算,
  - 選択,
  - 反復,
  - 入出力,
  - 記憶(変数),
  - 順序付け,
- 並びに、これらの組み合わせ

Ice代 = Iceの価格 × (1+0.05);

```
if (Ice代 > 400円) {  
    高すぎる!  
} else {  
    手頃な値段!  
}
```

# プログラムとアルゴリズム

- アルゴリズム： 実際の記述形式に拠らない解法の手順
- プログラム： アルゴリズムを，特定の記述形式（プログラミング言語）で書き表したものの



## 4. なぜプログラミングを学ぶのか？

### ■ 人間の労力を省くため

問題解決の方法を学び計算機に仕事(計算)をさせる.

- ◆ 大規模計算(風力, 弾道, 圧力, 熱量..), 変換(暗号化, 翻訳..)
- ◆ 検索(図書検索, Web検索..), 分類, 通信

プログラミングに要する時間と, 人間が自分自身で処理する時間とのトレードオフを考える!

### ■ 人間の思考過程や計算機の仕組み, 計算の手順などを理解するため

- 人間は, 物体や事象, 問題を, どのように認識し, 理解し, どう考えているのか? また, 問題を解決するために, 何をしているのか?
- 計算機は, どのようにして, 命令を実行しているのか?

# 5. Xcodeが自動生成するC言語のプログラム

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

```
int main(int argc, const char * argv[]) {  
    // insert code here...  
    printf("Hello, world!\n");  
    return 0;  
}
```

コメント:  
実行されない記述

式の後ろにセミコロン  
を付けて文にする。

記号	読み	意味
;	セミコロン	文(一部の文)の終わり
,	カンマ	記号の区切り
"	ダブルクォート	文字列の開始と終了
	空白	語の区切り
{	開き中括弧	ブロックの開始
}	閉じ中括弧	ブロックの終了

# main 関数

- プログラムの実行を開始すると、自動的に main 関数から実行される。
  - プログラムには、必ず main 関数が書かれていなければならない。
  - プログラムに複数の main 関数が書かれているとエラーになる。

前のページの例では、

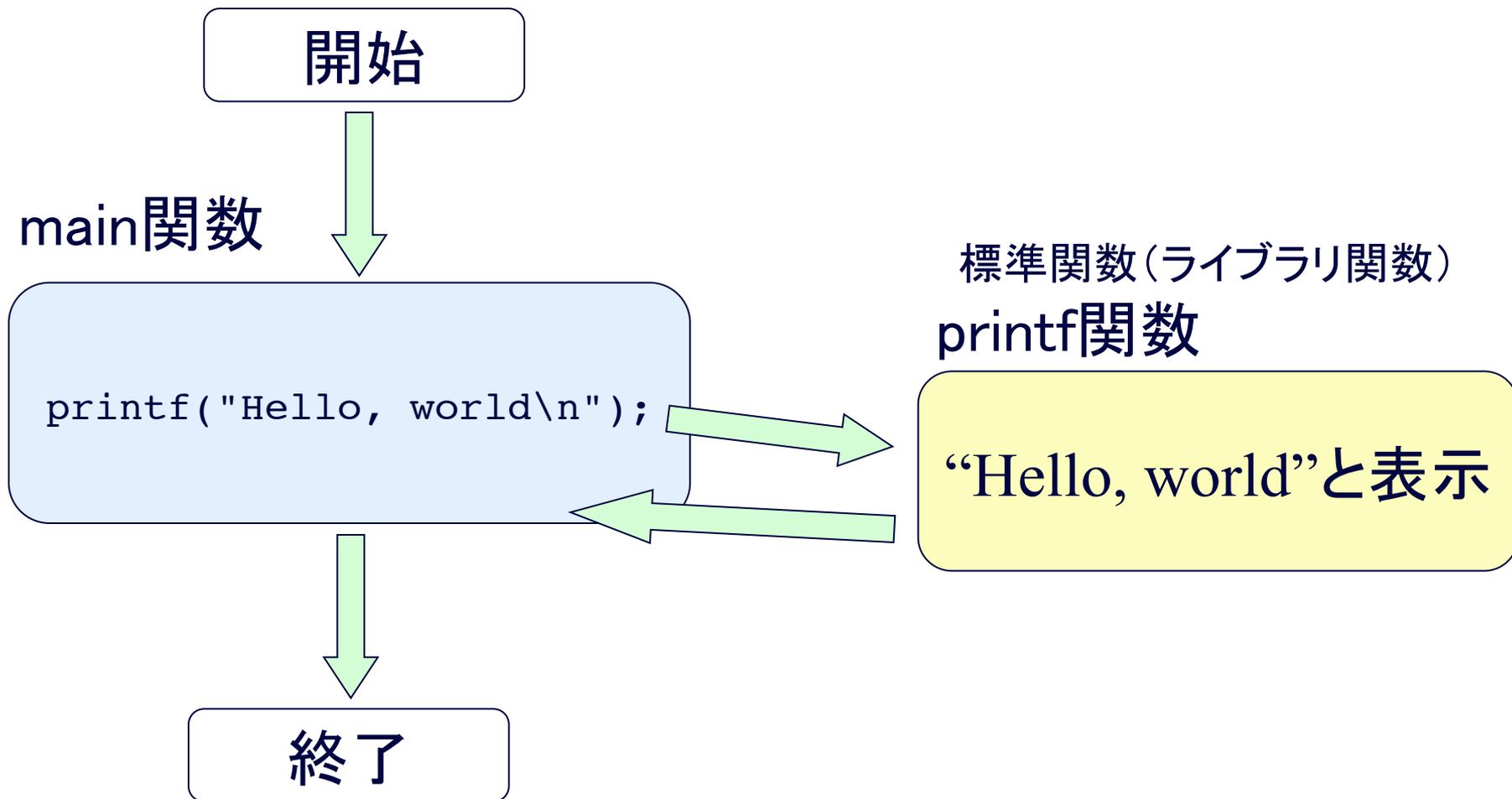
ここが main 関数

```
#include <stdio.h>

int main(int argc, const char * argv[])
{
    // insert code here...
    printf("Hello, world!\n");
    return 0;
}
```

この行から実行

# プログラムの流れ



# 標準関数とヘッダファイル

- 関数は呼び出す前に必ず定義する必要がある。
- 標準関数(ライブラリ関数)
  - システムが定義している関数
  - 定義はヘッダファイルに書いてあるので, 該当するヘッダファイルを読み込む(プリプロセス)だけで使える(#include).

## ■ ヘッダファイル

`stdio.h` 入出力

他にも

`math.h` 数学関数

`ctype.h` 文字クラス・テスト

`string.h` 文字関数

`stdlib.h` ユーティリティ関数(数値変換, メモリ割当等)

# プリプロセス

- プリプロセス行(#で始まる行)は, コンパイルする前に解釈され展開される.

`#include <stdio.h>`      `stdio.h` の内容を読み込む

`#define MAX 512`      以降, `MAX`を512に置き換える

# 書式付出力関数 printf

- 前もって, `stdio.h` を `include` すること
- プログラム中では下のように記述

```
printf("表示したい文字列\n");
```

引数(この内容が表示される)

- `\n` 改行
  - エスケープシーケンス  
(画面に表示されない)
  - その他のエスケープシーケンス

`\t` 水平タブ  
`\r` 復帰

バックスラッシュ(`\`)は option キーを押下した状態で「¥」キーを押すと入力できる.

## 6. Xcode統合環境の使い方

- 別の資料にまとめているので、そちらを参照して欲しい。

## 7. 今日の練習問題

Level	問題
C	ターミナルに「おはようございます(Good morning)」と表示するプログラムを作成せよ.
B	printf関数を4回使って「おはよう」「こんにちは」「こんばんは」「おやすみ」を各1行で表示するプログラムを作成せよ.
B	上のプログラムをprintf関数を一つだけ使って実現せよ.