計算機とアルゴリズム I 第3回

森 義之

ソースファイルについて

ソースファイル

- LaTeX の場合

ソースファイルの作成 ⇒ 変換 (コンパイル) ⇒ 表示可能ファイル

の流れで、数式等見やすい表示 (pdf ファイル作成) が可能

ソースファイル

- LaTeX の場合

  ソースファイルの作成 ⇒ 変換 (コンパイル) ⇒ 表示可能ファイル

  の流れで、数式等見やすい表示 (pdf ファイル作成) が可能

- C 言語の場合

  ソースファイルの作成 ⇒ コンパイルとリンク ⇒ 実行可能ファイル

  の流れで、コンピュータが実行できるファイルの作成が可能

ソースファイル

- この講義においてソースファイルのファイル名に使える文字

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz0123456789+-_( )

☆日本語, 空白, その他の記号や全角文字は使わない！

厳守! 環境によっては、上記以外の半角文字や日本語が使用可能なときもあるが、
この講義の提出用ファイルのファイル名には使用しないこと。

誤ったファイル名

・良いファイル名

`kadai-2-1.c`

`mondai0201.c`

・悪いファイル名

`課題A.c`	ファイル名に、全角文字 (日本語) を入れない
`kadai A.c`	ファイル名に、空白を入れない
`Ｋａｄａｉ.c`	ファイル名に、全角文字を入れない
`kadai2.1.c`	ファイル名に、ドットを入れない

ファイル名が正しくない場合、コンパイルエラーが出ることがある

ソースファイル(拡張子)

・ソースファイルの決まり

ファイル名の後半に拡張子 .c を付ける。

この .c も必ず半角

また、.c の後に文字を書いてはいけない。　例えば、.com とすると意味が変わる。

※ 拡張子とは「ファイルの形式を示す文字列」のこと。
※ 余談：拡張子には間に . を挟むファイルもある。有名なのは.tar.gz、LaTeX をコンパイルすると.synctex.gz ができる。また、 . から始まるファイルも存在する。

課題 A.1.

以下の (1) 〜 (10) において、C 言語のソースファイルのファイル名として
この講義で正しいものをすべて答えよ。

(1) `kadai_A_1.c`

(2) `katai.No2.c`

(3) `かだい001.c`

(4) `kadai_orz.c`

(5) `A1-(1)and(2).c`

(6) `toi1(1),(2).c`

(7) `No2025/5/10.c`

(8) `kadai-2-1.cpp`

(9) `(-_-)v.c`

(10) `kadai[1].c`

コンパイルとプログラムの例

コンパイルとは

・復習

ソースファイルの作成 ⇒ コンパイルとリンク ⇒ 実行可能ファイル

例 (ソースファイル)

```
#include <stdio.h>
int main(void)
{
    printf("Hello, C world!¥n");
    return 0;
}
```

最初のCプログラム

例 (Hello C と表示するプログラム)

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello C¥n");
5     return 0;
6 }
```

実行結果

```
Hello C
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello C¥n");
5     return 0;
6 }
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello C¥n");
5     return 0;
6 }
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello C¥n");
5     return 0;
6 }
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello C¥n");
5     return 0;
6 }
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void){
3     printf("Hello C¥n");
4     return 0;
5 }
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void)
3     { printf("Hello C¥n");
4     return 0;
5 }
```

最初のCプログラム

```
1 #include <stdio.h>
2 int main(void)
3 {
4     printf("Hello C¥n");
5     return 0;
6 }
```

課題 A.2.

以下の問題のプログラムにおいて、誤りをすべて答えよ。

問題のプログラム

```
♭include <studio.h>
int main(void)
{
   printf("Hello C¥n");
   printf("World¥n");
   return 0:
}
```

※　実際のプログラムなので、行番号は記載していない。

プログラムを読みやすくする

改行は不要？

例 (2.1.1)

```
#include <stdio.h>
int main(void)
{
    printf("Hello, C world!¥n");
    return 0;
}
```

例 (2.1.1 改変)

```
#include <stdio.h>
int main(void){printf("Hello C world!¥n");return 0;}
```

改行は必要？

例 (2.1.1 さらに改変)

```
#include <stdio.h>
int main
(void)      {
   printf
   (
       "Hello C¥n"
   );
   return 0;
}
```

字下げ(インデント)

例 (インデント有り)

```
#include <stdio.h>
int main(void)
{
   printf("Hello C¥n");
   return 0;
}
```

例 (インデント無し)

```
#include <stdio.h>
int main(void)
{
printf("Hello C¥n");
return 0;
}
```

※ 字下げ (インデント) 行うと、関数の定義内容が見易くなる？

空白類文字

プログラムを見易くするために使った

- 空白
- 改行
- 字下げ（インデント)

} C 言語のコンパイラは空白文字として扱う。

これらをまとめて以下、**空白類文字**という。

前記の通り、空白類文字はいくつ連続していも 1 つの空白と見なす。したがって、見易いように空白類文字を使ってよい。

質問 Q.1.

以下の、(ア) と (イ) でどちらが見やすいか、差はないか？

例 (ア)

```
year=1984,1986,1988,2001
month=3,8,4,7
day=11,2,16,20
week=sun,sat,sat,Fri
keyword=ohmu,baru,toto,hiro
```

例 (イ)

```
year    = 1984,1986,1988,2001
month   =    3,   8,   4,   7
day     =   11,   2,  16,  20
week    =  sun, sat, sat, Fri
keyword= ohmu,baru,toto,hiro
```

♠ 補足

空白 (の数) を黒板、ノート、プリントで明記する場合、␣を用いることがある。

※大きく表示させると⌊␣␣␣␣␣␣␣⌉である。

♠ 補足

空白 (の数) を黒板、ノート、プリントで明記する場合、␣を用いることがある。

※大きく表示させると⎵である。

例えば、

```
int     main    (     void      )       { int i,sum   ;
```

の空白の数を明記する場合

```
int␣␣␣␣main␣␣␣␣(␣␣␣␣void␣␣␣␣␣)␣␣␣␣␣␣{␣int␣i,sum␣␣;
```

のように書く。

コメント

プログラムを分かり易くするために、プログラムの中にコメントを書くことがある。

1 行のコメントの場合
//
を使い、複数行にまたがるときは
/*
.....
.....
*/
を使う。

コメント

例 (コメント)

```
 1 #include <stdio.h>
 2 /*
 3 このプログラムは、あるグラフ.....
 4 始点と終点に対して....
 5 */
 6 int main(void) {
 7     int s,t,u,v;  // 始点と終点用の変数
 8     double ds,du;  // 重み用変数
 9 //ここから開始
10     printf("Start¥n");

30 /* 1 行のコメントをこのように書くこともできる*/

80 }
```

プログラムのルール

トークン

☆トークンの種類

・キーワード	予約済みの名前
・識別子	データの位置などを識別するための名前 例えば関数名
・定数	固定的な値、例えば 1 や 0.5 などの数値
・リテラル文字列	二重引用符"で囲まれた文字列
・演算子	計算に用いる記号、例えば +や - など
・区切り子	[ ] ( ) { } ; , #など

トークン

☆トークンの種類

・キーワード	予約済みの名前
・識別子	データの位置などを識別するための名前
	例えば関数名
・定数	固定的な値、例えば 1 や 0.5 などの数値
・リテラル文字列	二重引用符"で囲まれた文字列
・演算子	計算に用いる記号、例えば +や - など
・区切り子	[ ] ( ) { } ; , #など

課題 A.3.

以下の問題のプログラムにおいて、誤りをすべて答えよ。
誤りは、行番号とその行の間違いの組として答えよ。

問題のプログラム

```
1 #include <stdio.h>
2 int main(void){
3   {printf("Hello C");
4   printf" World!!¥n");
5   {
6     printf("Sample File¥n");
7  }
8  return 0;
9 }
```

行番号、エラー、関数、printf 文

- 行番号：既に説明済み。間違える人が多かったので再度
- エラー：エラーが出てからで良い、ここを確認
- 関数：雰囲気を知る
- printf 文：対面講義で

名前 (識別子)

プログラム中で使用する名前 (変数名、関数名、記号定数名) で使用できる文字には決まりがあり、以下の半角文字に限る。

名前 (識別子) に使用可能な文字

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 _

名前 (識別子)

さらに、名前 (識別子) の付け方には以下の決まりがある。

- **英文字 または _ で始めなければならない。**
- **長さに制限はない。**
- **予約語を識別子に使うことは出来ない。**
  例えば、変数名に for を使うことは出来ない。
- **大文字と小文字は区別される。(max と Max は異なる識別子)**

予約語

予約語

auto	double	int	struct	break	else
long	switch	case	enum	register	typedef
char	extern	return	union	const	float
short	unsigned	continue	for	signed	void
default	goto	sizeof	volatile	do	if
static	while				

※ 予約語は名前として使うことは出来ない。

予約語

☆ プログラムで include したヘッダファイルに定義されている名前も使用できない。

★ 例) `stdio.h` で読み込んだ `printf` や `scanf` などは使えない。

☆ さらに、rand や srand は現状では使用できるが、`strlib.h` を読み込むと使えなくなる (予約語になる)。

課題 A.4.

以下の (i)〜(xii) に対して、名前 (識別子) として使用できるモノをすべて答えよ。ただし、include される (`stdio.h` も含めて) ヘッダファイル内にある名前は考えなくてよい。

(i) `Hensu`	(ii) `2ban`	(iii) `No3`
(iv) `number 4`	(v) `A4(v)`	(vi) `min`
(vii) `quattro`	(viii) `Kadai_No`	(ix) `For`
(x) `Nintendo`	(xi) `switch2`	(xii) `hoshii--`

定数

定数

C 言語で扱われるデータ
- 定数
  - 整数定数
  - 浮動小数点定数
  - 文字定数
  - *etc.*
- 変数

例 (定数)

```
dt=100;
i=100;
x=3.141592;
syokichi=-0.01234;
```

♠ **補足** =は"等しい"という意味ではなく、代入を意味する。

整数定数

整数定数は小数部のない数値定数である。
また、整数定数には 10 進数,8 進数,16 進数の表記や、符号の有無を示す接尾語 (u, U) がある。

例 (a に 30 を代入する例)

```
a=30;
a=036;
a=0x1e;
a=30u;
```

♠ **補足** 環境によっては定数の大きさを表す接尾語 (l,L,LL) が利用できる。
このような、データ型については次回。

浮動小数点定数

浮動小数点定数は小数部を持った数値定数である。また、e や E を用いて指数形式で表す。

例 (a に 123.456、b に 0.000123 を代入する例)

```
a=123.456;
a=1.23456e+02;
b=0.000123;
b=1.23e-04;
```

♡ **注意!** 0.00000000000123 などは指数形式を用いないと定義できない。

文字定数と文字列リテラル

☆ 文字定数は、シングルクォーテーションで囲み (例えば 'A')、その文字のもつ文字コードを表し、'A' は文字 A の文字コードを示す。

例 (以下の (1) と (2) は同じ意味である。)

```
char c;
c='A';  // (1)
c=65;   // (2)
```

☆ 文字列リテラルは、ダブルクォーテーションで囲まれた文字列のことである。文字定数も本質的に定数である。(例："moji ")

♡ **注意!** 'A' と"A"は異なるモノ！

エスケープシーケンス

例 2.1.1. の `printf()` 関数でみた `¥n` は改行を意味する文字である。この文字は先頭に `¥`があるため、本来の文字 `n` を離れ (エスケープし)、特別な意味の文字となる。このような文字を**エスケープシーケンス**と呼ぶ。
よく使うエスケープシーケンスは、

`¥n` (改行), `¥t` (水平タブ)

である。
詳しくは、実際にプログラムを作成したときに説明する。

記号定数, 関数マクロ, const 付き変数

本日の課題

・課題 A.1. 〜 A.4
・質問 Q.1.

提出方法はメールで mori@xmath.ous.ac.jp まで送る。
件名は学生番号と課題 A 質問 Q と書き、

例) S24M999 課題 A. 質問 Q

メールの本文に解答を書くこと。

締め切りは 5 月 16 日 23:59。