

# 物理学情報処理演習

## 4. C言語と編集

ver20160510\_2

2016年5月10日

4.1 プログラム開発

4.2 C言語の構造

4.3 入出力

### 参考文献

- やさしいC++ 第4版 高橋 麻奈 (著)  
ソフトバンククリエイティブ
- プログラミング言語C++第4版  
ビャーネ・ストラウストラップ, Bjarne Stroustrup, 柴田 望洋
- Numerical Recipes: The Art of Scientific Computing, Third Edition in C++

身内賢太朗

レポート提出: [fsci-phys-jouhou@edu.kobe-u.ac.jp](mailto:fsci-phys-jouhou@edu.kobe-u.ac.jp)

# 4.1 プログラム開発

- プログラム開発の流れ(先週の作業を思い出しながら。)

- 4.1.1 プログラム編集

- ソースコードファイルを作る

- 4.1.2 コンパイル

- 文法エラーをチェック

- 4.1.3 デバグ

- ソースコードとコンパイルメッセージを照らし合わせて誤りを発見する。

- 実行

- 実行時エラーをチェック
- 実行結果をチェック

```
$mkdir lesson4  
lesson4を作成
```

ブラウザで本日のページを開き、hello.cxxを右クリック「リンク先のファイルを別名で保存」する。拡張子txtはつけない。場所は各自のアカウント名の下のlesson4を選択する。

```
$cd ~/lesson4  
lesson4に移動
```

```
$ls  
hello.cxxが存在することを確認
```

```
$g++ -o hello hello.cxx  
コンパイル
```

```
$ls  
helloが存在することを確認
```

```
$/hello
```

## 4.1.1 プログラム編集

- hello.cxx を眺める。

```
$cd
```

```
$cd lesson4
```

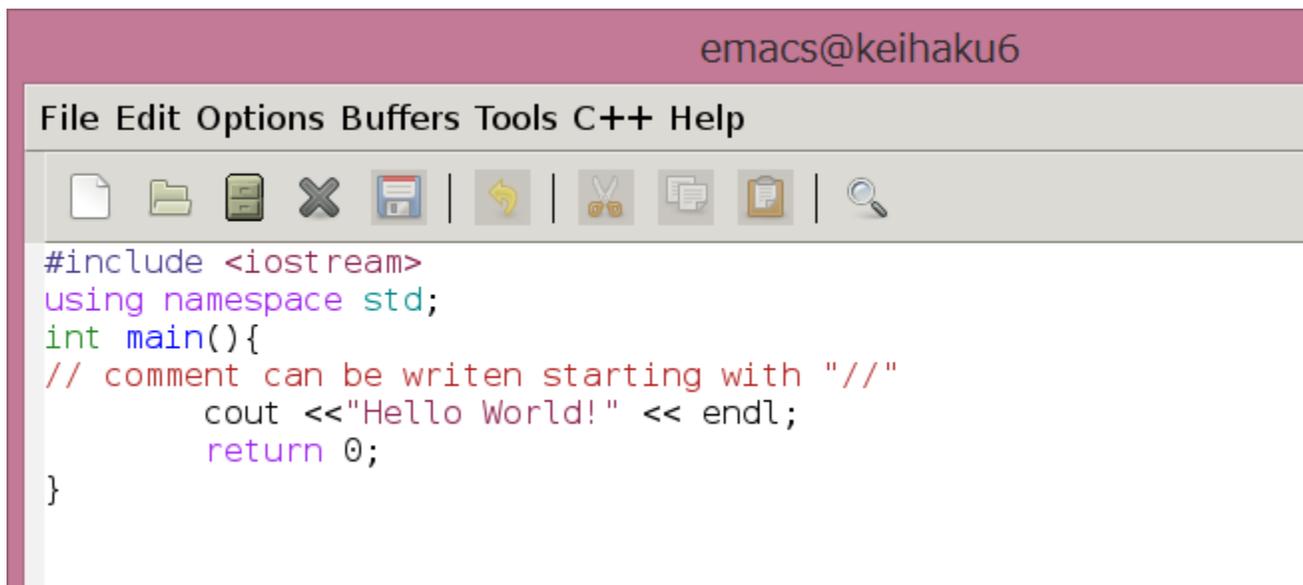
```
$cat hello.cxx
```

表示

```
$emacs hello.cxx &
```

editorを用いて hello.cxxを編集

editor: テキストファイルの編集ソフト。純粋にテキストファイルをいじるための機能に特化している。書式情報などは保存、表示されない。



```
emacs@keihaku6
File Edit Options Buffers Tools C++ Help
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "//"
    cout << "Hello World!" << endl;
    return 0;
}
```

### 演習4.1.1 コマンド集

[http://ppwww.phys.sci.kobe-u.ac.jp/~miuchi/education/lecture/2016\\_jouhou/2016\\_jouhou\\_commands.pdf](http://ppwww.phys.sci.kobe-u.ac.jp/~miuchi/education/lecture/2016_jouhou/2016_jouhou_commands.pdf)  
を見ながらやってみよう(提出は不要)

emacs の ウィンドウ分割、戻す

別file名で保存 複数fileを開く バッファ間の移動

行の先頭に移動 → カーソルより右をカット → 別の場所にペースト

文字列検索 などなど。

## 4.1.2 コンパイル

- GNUのコンパイラ

- gcc: Cコンパイラ

- g++: C++コンパイラ

本演習ではC++を使用しますが、クラスを利用したオブジェクト指向というC++特有の機能まではカバーしません。C++でのみ許されている若干の文法表現のみ使用します。

- i. pre-processor

- ii. C-compiler

- iii. assembler

- iv. linker

### 全てを含んでいる

識別子により判断

.c: C言語ソース

i, ii, iii, iv

.cxx .C .cpp : C++言語ソース

i, ii, iii, iv

.h: ソース(ヘッダーファイル)

i, ii, iii, iv

.I: プリプロセス後のC言語ソース

i, ii, iii, iv

.o: オブジェクトファイル

iv

\$ls

hello.cxxが存在することを確認

```
$g++ -o hello hello.cxx  
コンパイル
```

\$ls

helloが存在することを確認

\$/hello

〜

\$ls

hello.cxxが存在することを確認

```
$g++ -o hello hello.cxx  
コンパイル
```

\$ls

helloが存在することを確認

\$/hello

## • g++ のオプション

- 書式 `g++ [option | filename]...`

### よく使うオプション

- `-c`: コンパイルまたはアセンブルまでで止める。コンパイラ  
の出力はそれぞれのソースファイル(`xxx.cxx`)に対応したオブ  
ジェクトファイル(`xxx.o`)となる。
- `-o file`: 出力先を`file`に指定  
`-o` と `file`の指定がないと `a.out` という実行fileができる。
- `-V`: バージョン情報
- `-I dir`: `dir`をインクルードファイルの検索するディレクトリのリス  
ト中に追加
- `-O1, -O2 ...`: 最適化を行う。数字が大きい方が最適化が  
深い

## 4.1.3 デバッグ

- 文法間違いなどがあると、コンパイル時にメッセージ
  - error : 重大な間違いでコンパイル未完了。要debug
  - warning : 軽微な間違いなどでコンパイルは完了したが、想定通りに動作しない可能性あり。 → debug推奨
- コンパイルは通るが、実行時にエラー

# • コンパイル時のメッセージ例 (よくある順。必要に応じて確認のこと。)

## <Error>

- parse error before ‘xxx’  
品詞解析エラー。括弧、セミコロン、クォーテーションのつけ間違いで、文の構造がおかしくなっているとき
- ‘xxx’ undeclared  
宣言せずに変数または型‘xxx’が使われたとき
- redeclaration of ‘xxx’  
変数‘xxx’を2回宣言したときinvalid value in assignment  
定数等、代入できないものに値を代入しようとしたとき。
- invalid operands to binary +  
2項演算子 ‘+’ の使い方がおかしいとき  
例： x = x + “yy” ;double とchar\*は足せない
- array subscript is not an integer  
配列の添字が整数でないとき
- conflicting types for ‘xxx’  
関数‘xxx’の宣言部と定義部で戻り値の型、引数の個数・型が違うとき
  
- too many arguments to function ‘xxx’
- 関数‘xxx’の引数の個数が多いとき
- incompatible type for argument i of “xxx”
- 関数‘xxx’のi番目の引数の型が違うとき
- undefined reference to ‘xxx’
- 関数‘xxx’が定義されていないとき
- 定義がされていない、または関数名のミスタイプ
- ライブラリーがリンクされていない

## <Warning>

- warning: assignment makes pointer from int without a cast  
整数をポインター型の変数に代入したとき
- warning: assignment from incompatible pointer type  
違う型へのポインター型の変数に変換したとき
- warning: control reaches end of non-void function  
戻り値が‘void’以外なのにreturn文が無いとき
- warning: unused variable ‘xxx’  
変数を宣言したのに使わなかったとき

# • 実行時のメッセージ例

(よくある順。必要に応じて確認のこと。)

## <Error>

- Segmentation fault (core dumped)
  - 保護されているメモリー領域(コード領域、カーネル領域等)にアクセスを行った (SEGVとも書き表す)
- nan
  - 負の数のsqrtを求めるなど、実行できない演算を行った。
- inf (または -inf)
  - 0で割るなど無限大の演算を行った。
- Illegal instruction
  - 不正な命令の発行
- Bus error
  - ワード境界を超えてアクセスした。または外部バスのエラー

## 4.2 C言語の構造

- hello.cxx を眺める。  
\$cat hello.cxx

```
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "//"
    cout <<"Hello World!" << endl;
    return 0;
}
```

## 4.2 C言語の構造

- hello.cxx を眺める。  
\$cat hello.cxx

### 4.2.1 関数

```
#include <iostream>
using namespace std;
```

```
int main(){
```

```
// comment can be written starting with "///"
    cout <<"Hello World!" << endl;
```

```
    return 0;
```

```
}
```

hello.cxx

- 関数
  - 文の集合で、特定の機能を持つ。
  - 型に応じた値を返す(return)。
  - {}で囲まれる。
  
- プログラムは関数の集まりである。
- 実行モジュールは必ずmain関数を持つ。

## 4.2.2 ライブラリ

```
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "//"
    cout <<"Hello World!" << endl;
    return 0;
}
```

hello.cxx

- ライブラリ: 特定の目的を持った関数の集合。  
#includeで読み込む
  - <iostream> 入出力に関する関数
  - <stdlib.h> 基本的なライブラリ
  - <string. h> 文字列を扱う関数
  - <math. h> 数学関係の関数

## 4.2.3 文

```
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "//"
    cout <<"Hello World!" << endl;
    return 0;
}
```

hello.cxx

- 文
  - ;までが一文
  - 宣言文と実行文がある
    - 宣言文 変数の宣言
    - 実行分 式、制御

## 4.2.4 書式

```
#include <iostream>
using namespace std;
int main(){
```

hello.cxx

```
// comment can be written starting with "//"
```

```
    cout <<"Hello World!" << endl;
    return 0;
```

```
}
```

- インデント

- 同じ階層の文は揃えて記述する。
- TABで自動インデント可能

コメントは超重要。

- コメントアウト

- 行の中で //以降は コンパイル時に無視される。
- /\*\* と \*\*/ で囲む表記も可能。複数行にまたがるコメントも可能。

プログラムを書き換えるとき、  
もともとの記述をコメントアウトしておくとすぐに元に戻せる。

## 4.3 入出力

- hello.cxx を眺める。  
\$cat hello.cxx

```
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "/"
    cout <<"Hello World!" << endl;
    return 0;
}
```

# 4.3 入出力

## 4.3.1 出力

```
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "/"
    cout <<"Hello World!" << endl;
    return 0;
}
```

hello.cxx

- cout 標準出力
  - `iostream` の `std` という namespace に定義されている。
  - 標準出力に出力するための関数。
  - コマンドラインでのリダイレクション `>` 等でファイルに書き込める。
- cerr 標準エラー出力
  - 常に画面に表示される

参考file:hello\_hello.cxx

### 演習5.3.1 (提出は不要)

hello.cxxの名前を変えてコンパイルしてみよう。実行ファイルの名前も適当に変えること。

出力する文字を変えてみよう。元の出力行はコメントアウトしておくこと。

coutとcerrを共存させ、ファイルへとリダイレクトしてみよう

## 4.3.2 入力

```
#include <iostream>
using namespace std;
int main(){
// comment can be written starting with "///"
char c[16];
cout <<"Hello, input a message >";
cin>>c;
cout <<c << endl;
return 0;
}
```

hello\_cin.cxx

- cin 標準入力
  - 標準入力から入力するための関数。
  - ここではcという変数に入力している。

参考file:hello\_cin.cxx

演習4.3.2 (提出は不要)  
hello\_cin.cxxをダウンロード、実行してみよう。

## 4.3.3 コマンドラインパラメータ

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]) {
// comment can be written starting with "//"
    cout << "Input a message with a command." << endl;
    cout << "Number of command line parameters (argc)=" << argc << endl;
    cout << "Comand line paremeter 0 (argv[0]) : " << argv[0] << endl;
    cout << "Comand line paremeter 1 (argv[1]) : " << argv[1] << endl;
    return 0;
}
```

hello\_com.cxx

- main関数は引数を取ることができる。
  - 引数:実行時に コマンド名に続けて与えるパラメータ
  - argcにパラメータの数、argvには引数の内容が代入される。

参考file:hello\_com.cxx

演習4.3.3 (提出は不要)  
hello\_com.cxxをダウンロード、実行してみよう。

実行時にパラメータを渡してみよう。

\$/hello\_com

\$/hello\_com hello

\$/hello\_com hello hellooo

の出力の違いを比べてみよう

## 課題4:C言語と編集

以下の仕様を持つプログラムを製作し、ソースコード及び出力結果を提出せよ。

- ① コマンドライン引数として名前を入力すると

Hello *名前*, input your student number.>  
と出力されるようにする。

- ② そこに学生証番号を入れると(cinを使う)

Thank you.

Your name is *名前*.

Your student ID is *学生証番号*.

と出力されるようにする。

上記で*名前*と*学生証番号*は入力した文字列が代入されるようにする。

# 課題提出

- 宛先 [fsci-phys-jouhou@edu.kobe-u.ac.jp](mailto:fsci-phys-jouhou@edu.kobe-u.ac.jp)
- 件名 2016-report04\_学籍番号の下4桁
- 本文 学籍番号と名前
- 添付ファイル:
  - 2016\_jouhou\_04\_学籍番号の下4桁.cxx
  - 2016\_jouhou\_04\_学籍番号の下4桁.txt
- 締め切り 2016年5月17日(火)13:00