物理学情報処理演習

2017年5月2日

ver20170411

4. C++言語① 開発・編集

4.1 プログラム開発

4.2 C言語の構造

4.3 入出力

身内賢太朗

レポート提出：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

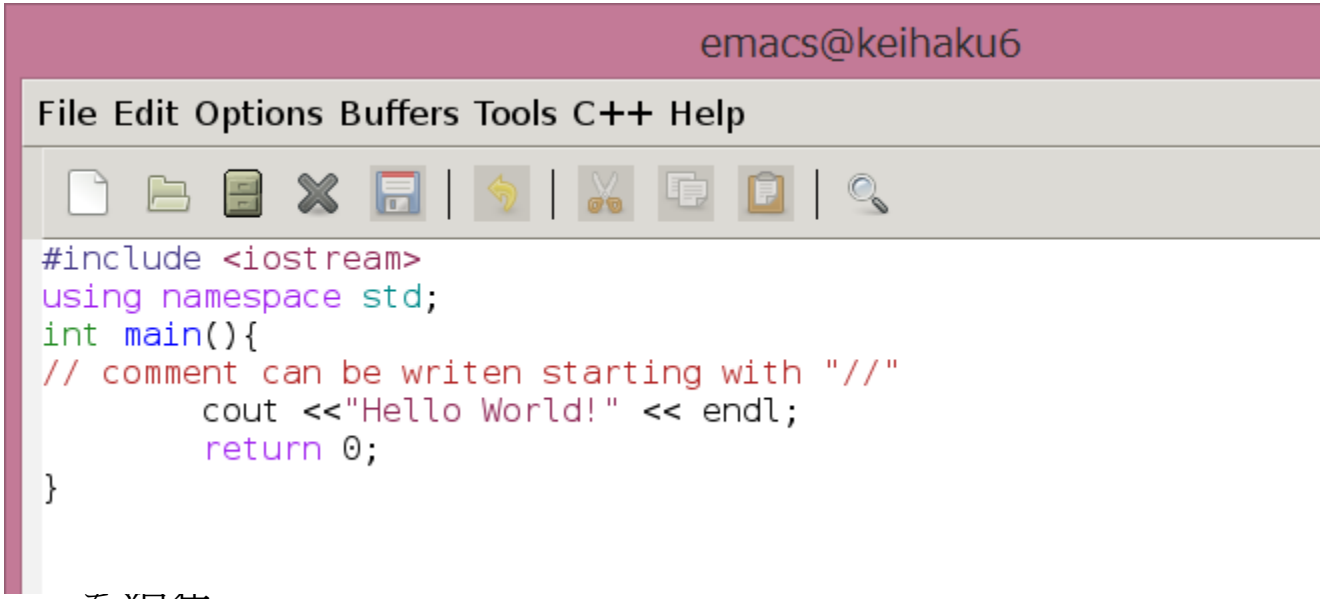
$ls
　　hello.cxxが存在することを確認

$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。

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
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　バッファー間の移動

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

$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

- 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

コピー

```
$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。）

〈Erorr〉

- 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
- redeclearation 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返り値の型、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
  返り値が’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。）

〈Erorr〉

•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でる演算を行った。

•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 writen 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

hello.cxx

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

- 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

hello.cxx

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

- 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

hello.cxx

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

- 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

hello.cxx

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

4.3 入出力

4.3.1 出力

hello.cxx

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

- 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

hello_cin.cxx

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

- 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

hello_com.cxx

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[]){
// comment can be writen 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;
}
```

- 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

ソースコードファイル名：2017_jouhou_04_学籍番号の下4桁.cxx
出力結果ファイル名：2017_jouhou_04_学籍番号の下4桁.txt

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 プログラムを実行するとすると
　Hello, input your name.>
と出力されるようにする。

② そこに名前を入力すると(cinを使う)
　Hello*名前*, input your student number.>
と出力されるようにする。

③　そこに学生証番号を入力すると(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
- 件名　2017-report04_**学籍番号**の下4桁
- 本文　学籍番号と名前
- 添付ファイル：
  - 2017_jouhou_04_学籍番号の下4桁.cxx
  - 2017_jouhou_04_学籍番号の下4桁.txt
- 締め切り　2017年5月9日（火）13：00