物理学情報処理演習
8. C言語④
2015年6月12日
ver20150612

本日の推奨作業directory

lesson08

8.1　文字列

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

課題提出期限　2015年6月19日13:00

8.1 文字列

- 8.1.1文字列処理
  - ある変数に対するアドレス(メモリ上の位置)をもつ変数
  - N個の文字からなる文字列は、N+1個の大きさの文字配列
  - 末尾の文字は、NULL文字('¥0')
    - 例　　char c[16];

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

hello_2.cxx

配列cの中身

演習8.2.1(提出不要) hello_2e.cxx、hello_2ee.cxx、hello_2eee.cxx をdebugしてみよう。エラーメッセージをよく確認すること。

- 8.1.2 文字列処理
  - 標準ライブラリの<string.h>に定義してある文字列処理ルーチン

char* strcpy(char* s, const char* ct)	NULL文字を含めて文字列ctを文字列sにコピーし、sを返す。
char* strncpy(char* s, const char* ct, int n)	文字列ct内からn文字を文字列sにコピーし、sを返す。ctがn文字より少ないときはNULL文字をつめる。
char* strcat(char* s, const char* ct)	文字列ctを文字列sの終わりに連結し、sを返す。
char* strncat(char* s, const char* ct, int n)	文字列ct内から最大n文字を文字列sの終わりに連結し、sを返す。
int strcmp(const char* cs, const char* ct)	文字列csと文字列ctを比較する。一致していれば0を返す。一致していなければ、相違が発見された文字どうしの数値上の差を返す。
size_t strlen(const char* cs)	文字列csの長さを返す。

演習8.2.2 (提出不要) hello_3.cxxを実行してみよう。内容を確認して、上記機能を試してみよう。

- 8.1.2 文字列処理
  - 標準ライブラリの<stdio.h>の関数sprintfも使える。

int printf (const char *format, . . .);	第一引数に書式を書き、第二引数以降に実際に出力される変数を書く。 出力は標準出力。
int sprintf (char *str, const char *format, . . .);	第一引数に代入されるべき文字、第二引数に書式を書き、第三引数以降に実際に出力される変数を書く。

hello_4.cxx

```
#include <iostream>
#include<string.h>
#include<stdio.h>
using namespace std;
int main(){
// comment can be writen starting with "//"
  int i;
  char c1[16]="Hello ";
  char c2[16]="World!";
  char c3[16];
  char c4[64];
  cout <<c1 << endl;
  strcat(c1,c2);
  cout <<c1 << endl;
  for(i=0;i<10;i++){
    sprintf(c3,"%d",i);
    //substitute i to c3 as a character
    sprintf(c4,"%s%s_%d.dat",c1,c2,i);
    //sprintf can have more than one paramters
    printf("%d¥t%s",i,c3);
    //same as cout << i << "¥t" << c3 << endl;
  }
  return 0;
}
```

<書式の例>
☆☆☆　%d　10進数
☆☆☆　%lf double
☆☆☆　%e e 指数表示
☆☆☆　%s 文字列
☆☆☆　フィールド幅 %とdなどの間に数字を入れる。
%3dで3桁で整数をs桁で表示
%2.1fで全体で2桁、小数点以下1桁
%2.1eで全体で2桁、小数点以下1桁
☆☆　%c 1文字

<エスケープシーケンスの例>

☆☆☆	¥n	改行
☆☆☆	¥t	タブ
☆☆	¥r	キャリッジリターン

演習8.2.2 (提出不要) hello_4.cxx, hello_5.cxxを実行してみよう。内容を確認して、上記機能を試してみよう。

- 8.1.3 多次元配列

任意の型の配列を多次元化することが可能。

fact_13.cxx

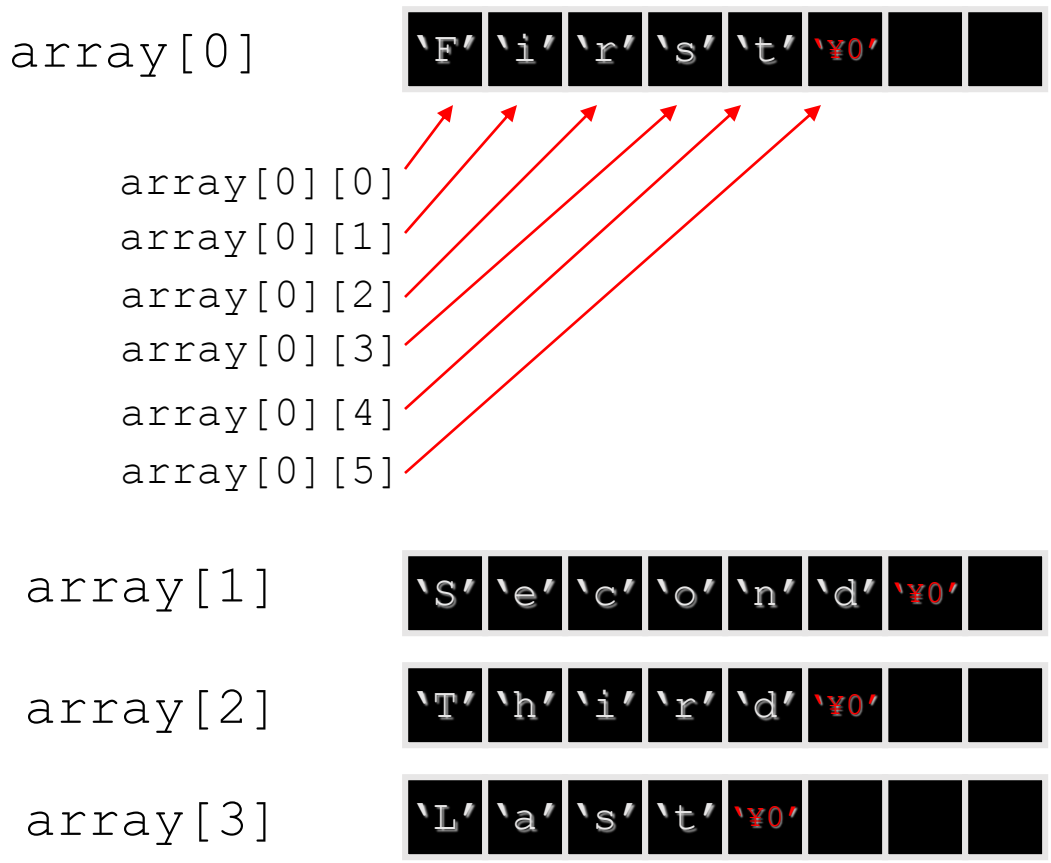
```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
using namespace std;
int fact(int n){
  //calculated the factorial
  int i,ans;
  ans=1;
  for(i=1;i<=n;i++){
    ans=ans*i;
  }
  return ans;
}
int main(int argc, char *argv[] )
{
  int n,max,i;
  int ans[2][20];
  if(argc>1){
    max=(int)atof(argv[1]);
  }
  else{
    // for default
    max = 10;
  }
  for(i=0;i<max;i++){
    ans[0][i]=i;
    ans[1][i]=fact(i);
  }
  for(i=0;i<max;i++){
    cout <<ans[0][i]<<"!="<<ans[1][i] << endl;
  }
    return 0;
}
```

演習8.2.3 (提出不要) 階乗計算のプログラムを多元配列を用いて書いたサンプルコード、fact_13.cxxを実行してみよう。出力3列目に1列目の数の2乗を出力するように変更を加えてみよう。

複数の文字列も多次元配列

`char array[A_SIZE][STR_SIZE]` が使える。

例　char array[4][8] = { “First”, “Second”, “Third”, “Last” };



演習8.2.3 (提出不要) hello_3.cxxを二次元配列を用いて書いてみよう。

- 出力　ofstream
  - これまでcout, cerrで出力していたが、出力先をファイルとすることも可能。

ofstest_1.cxx

```
#include <iostream>
#include <fstream>
#include<string.h>
#include<stdio.h>
using namespace std;
int main(){
  int i;
  char fout[16]="test.out";
  char c1[16]="Hello World!";
  ofstream ofs(fout);
  ofs << c1<<endl;
  return 0;
}
```

演習8.2.4 (提出不要) ofstest_1.cxxを書き換えて、出力ファイル名を変更してみよう。

レポート提出：fsci-phys-jouhou@edu.kobe-u.ac.jp
提出期限　2015年6月19日13:00

ソースコードファイル名：2015_jouhou_08_学籍番号の下4桁.cxx
出力ファイル名：2015_jouhou_08_学籍番号の下4桁.txt
出力画像ファイル名：2015_jouhou_08_学籍番号の下4桁.pdf

課題8：課題6のソースもしくは階乗を計算するサンプルコードfact_13.cxxを基にして、以下の様なコードを作成せよ。
提出はソースコード、gnuplotの出力ファイル、作業directoryでlsした出力をtxtとしたファイルの3点とする。
(プログラム仕様)
① 期待値μのポアソン分布で実現値Xを得る確率を計算、出力1列目にX（0から10）、2列目に確率をファイルに出力する。
② プログラム内で期待値μを0から10まで0.1刻みで変更し、それぞれのμに対応する出力ファイルをpoi_?.?.dat(例：μ=2.5の出力はpoi_2.5.dat)として、プログラムから自動的に生成されるようにする。（hello_5.cxx, oufstest_1.cxx参照。全部で100ファイルできるはずである。lsの出力をテキストファイルに落として提出のこと。）
③ ②で作成したデータのうちでμ=1,3,5について同じplotに描画せよ。（必要であれば、show.pltを使用せよ。）