★0　オリエンテーション

★0.1　この科目について

今年度は，前半の授業を高橋が担当（第８回までの予定）し，後半を樋口先生が担当します．

講義概要

計算機プログラムは，様々な操作を高速・効率的に処理したいという動機や目的があって作成されるものです．したがって，実用的なプログラムを作成する際には，現実の様々なデータ形式に応じた計算や入出力を行う必要があります．この科目では，C 言語を用いて，画像データやネットワークデータなどの具体的なデータを扱ったプログラムの作成を通して，より実際的なプログラミングの作法を学びます．

到達目標

現実の多様なデータを処理するためのプログラミング作法を修得する．少し大きな規模のプログラムソースを管理・作成できるようになる．

系統的履修

「プログラミング及び実習」の後を継ぎ，プログラミング能力のさらなる強化を目指します．

成績評価の方法

高橋担当分では，授業中に出題する課題の達成度で成績評価を行います (☆ 1)．後半の成績評価法については樋口先生におたずね下さい．

☆ 1) 小テストと演習

テキストと参考文献

英語の読み書きを学ぶために辞書が必要なように，プログラミング言語を学ぶためにはその書き方をすぐに調べられる情報源を手元に用意しておくべきです．書店に行けば山ほど C 言語の書籍がありますので，辞書のように使える&自分の好みに合うものを手に入れておくことを強くおすすめします．もちろん既に所有しているものが使えそうならそれでも構いません．使えそうなウェブページなどを自分で見つけておくのもよいでしょう．

いろいろ

- 真剣に授業に参加している人の邪魔をする行為（おしゃべり，途中入退室など）は禁止．
- 大学の授業は，授業時間の他にも自学自習することを前提に作られています．授業時間以外にも勉強することが必要です．
- やむを得ない理由で欠席した場合，有効な欠席届をすみやかに（次回の演習時などに）高橋に手渡してください．欠席した回の小テストの点数は除外して成績評価を行います．また，プログラミングの実習課題については，締切を延長することがあります (☆ 2)．ただし，欠席した回の授業内容については自習しておく必要があります（あたりまえのことですが）．小テストでは，前回の授業の範囲からの出題も当然ありますよ．

☆ 2) 詳しいことは高橋にお尋ねください．

★0.2 アクセス

以下は，前半分の授業に関する情報です.

この科目に関するウェブサイト

(1) 高橋のウェブページ (☆ 3)

`https://www-tlab.math.ryukoku.ac.jp/wiki/`

から「AProg/20XY」(XY は実際には年度を表す 2 桁の数字）をたどると，この科目のページにたどりつけます.

☆ 3) 高橋のページにたどりつくには，この URL をブラウザに直接入力するかわりに，理工学部や数理情報学科のウェブサイトからたどったり検索したりする手もありますね. `www.math.ryukoku.ac.jp`

(2) 龍谷 Moodle（龍谷大学 e-Learning システム）上のこの科目のページ

このサイト上で小テスト形式の演習を行います.

このページにたどりつくには，上記のこの科目のページからリンクをたどるのがはやいでしょう (☆ 4). 途中で，全学認証のユーザ名とパスワードでのログインが求められます.

☆ 4) 龍大ポータルサイトからも行けますが，だいぶ遠回りかも

はじめてこのページへ行こうとすると，ログインした後に,「このコースへ登録するためには「登録キー」というワンタイムパスワードが必要です」といわれるでしょう.「登録キー」は，初回の授業でお教えします.

高橋の連絡先など

- 研究室: 1–508（または 1–602）`e-mail: takataka[at]math.ryukoku.ac.jp`
- 高橋の 2017 年度後期の週間スケジュールは以下の通りです. この科目に関する質問等がある場合は，以下の OfficeHour の時間帯に訪問してください. 最新の情報はウェブ上および研究室の前に掲出してあります.

	月	火	水	木	金
1		AProg(1)		(会議)	
2		AProg(1)	(学科会議)		
昼休み			(学科会議)		
3			FIP		OH/基礎演習 B(2)
4	SJS		(教授会)		Office Hour
5	SJS		(教授会)		

OH = Office Hour　(ほげ) — あったりなかったりなイベント
FIP, AProg, SJS は授業です　ここに示したもの以外に，会議・セミナ・出張等が不定期であります.

(1) 高橋の担当は 11 月はじめまで　(2) 4 講時は 1–542 にいます. Office Hour も兼ねてますので質問等歓迎します

★1　復習—関数，ポインタ，配列

★1.1　関数

● 関数の定義の仕方

戻り値の型 関数名 (引数の宣言)
{
　　変数宣言と文
}

- 引数なしの場合は 引数の宣言 には何も書かない
- 戻り値なしの場合は 戻り値の型 は void にすればよい
- 「変数宣言と文」の中で return 文を用いると，戻り値を指定して関数をぬける（return 文は複数あってもよい．戻り値なしならなくてもよい）

● 関数の呼び出し方

下記の例参照

● 関数のプロトタイプ宣言 (☆ 5)

厳密な説明ではないが，ソースファイルの上の方で,「下の方（または別のソース中）でこんな関数定義してるよ」ということをあらかじめ宣言しておくもの．関数定義本体の１行目と同じものを書いて; をつける」と考えてもよいかも．

☆5)［発展］プロトタイプ宣言を省略すると，コンパイラはその関数の戻り値は int と仮定する．また，関数の定義を，それが呼び出される箇所よりも先に（main より先に）書けば，プロトタイプ宣言は省略できる．

Q1. 以下の func01.c の中には main の他にもう１つ関数が定義されている．

1. その関数の名前は何か
2. その関数を定義しているのはこのソースプログラムのどの箇所か
3. その関数を呼び出している箇所はどこか
4. その関数にはいくつの引数があるか
5. その関数の戻り値の型は何か
6. その関数のプロトタイプ宣言はどこか

func01.c

```
 1  #include <stdio.h>
 2
 3  int func(int x);
 4
 5  int main(void)
 6  {
 7    int a, b, x;
 8
 9    scanf("%d", &a);
10    b = func(a);
11    x = func(b+3);
12    printf("%d %d %d %d\n", a, b, x, func(x));
13
14    return 0;
15  }
16
17  int func(int x)
18  {
19    return x + 1;
20  }
```

Q2. func01.c をコンパイルして実行する手順を説明しなさい (☆ 6)．

☆6) 龍大計算機室の Linux 環境を想定してます．

Q3. func01.c をコンパイルしてできる実行形式を実行 (☆ 7) すると，どんな動作をするか説明しなさい．

☆7) 毎度このように書くのは面倒なので，まぎらわしくない限りは省略して「func01.c を実行すると」のように書くことにする．

Q4. func01.c を次のように変更したい．どのように修正したらよいか示せ．

- キーボードからの入力として実数を受け付ける
- 関数 func は，引数に与えられた数に 3.14 を加えた数を返すようにする

func01.c

```
 1  #include <stdio.h>
 2
 3  int func(int x);
 4
 5  int main(void)
 6  {
 7    int a, b, x;
 8
 9    scanf("%d", &a);
10    b = func(a);
11    x = func(b+3);
12    printf("%d %d %d %d\n", a, b, x, func(x));
13
14    return 0;
15  }
16
17  int func(int x)
18  {
19    return x + 1;
20  }
```

Q5. 次のような仕様の関数 hoge() の定義を書きなさい．

- 関数名は hoge
- 引数は 1 つ．その型は int で，仮引数の名前は fuga とする
- 戻り値の型は double とする
- 引数で受け取った値に 0.5 を加えた値を返す

Q6. 次のプログラムを実行するとどのような動作をするか説明しなさい．

［発展］関数 MultipleOf3() の中身は，左のんのかわりに
MultipleOfN(x, 3);
とだけ書く手もありますね (そもそも MultipleOfN() があれば MultipleOf3() は不要かもしれませんが)．

func02.c

```
 1  #include <stdio.h>
 2
 3  void MultipleOf3(int x);
 4  void MultipleOfN(int x, int n);
 5
 6  int main(void)
 7  {
 8    int a, i;
 9
10    for(i = 0; i < 3; i++){
11      printf("整数を入力してね: ");
12      scanf("%d", &a);
13      MultipleOf3(a);
14      MultipleOfN(a, 5);
15    }
16    return 0;
17  }
18
19  void MultipleOf3(int x)
20  {
21    if(x % 3 == 0){
22      printf("3 の倍数\n");
23    }else{
24      printf("3 の倍数じゃない\n");
25    }
26  }
27
28  void MultipleOfN(int x, int n)
29  {
30    if(x % n == 0){
31      printf("%d の倍数\n", n);
32    }else{
33      printf("%d の倍数じゃない\n", n);
34    }
35  }
```

★ 1.2 関数とポインタ

Q7. 次のプログラムを実行するとそれぞれどのような出力が得られるか.

hoge01.c

```
 1  #include <stdio.h>
 2
 3  void hoge(int x, int y);
 4
 5  int main(void)
 6  {
 7    int a, b;
 8
 9    a = 7; b = 3;
10    hoge(a, b);
11    printf("%d %d\n", a, b);
12
13    return 0;
14  }
15
16  void hoge(int x, int y)
17  {
18    y = x + y;
19    printf("%d %d\n", x, y);
20  }
```

hoge02.c

```
 1  #include <stdio.h>
 2
 3  void hoge(int x, int *y);
 4
 5  int main(void)
 6  {
 7    int a, b;
 8
 9    a = 7; b = 3;
10    hoge(a, &b);
11    printf("%d %d\n", a, b);
12
13    return 0;
14  }
15
16  void hoge(int x, int *y)
17  {
18    *y = x + *y;
19    printf("%d %d\n", x, *y);
20  }
```

注： 以下には，説明の簡単化のためにごまかしてる所（番地のつけ方など）があります

● hoge01.c の場合

7,9 行目 変数 a のためにメモリ中のある場所（例えば 26 番地）が用意され，そこに 7 が格納される．b についても同様.

10,16 行目 main は，変数 a と b の**値**つまり 26,27 番地の内容を hoge に渡す．hoge はそれらを自分で用意した場所（例えば 123,124 番地）に格納し，x,y と呼ぶことにする.

18 行目 x の値と y の値を足したものを y に格納（124 番地が書きかわる）.

19 行目 x,y の値を出力（7 と 10 が表示される）

11 行目 hoge をぬけて main に戻ってきた．a,b の値は変わっていない．それらを出力（7 と 3 が表示される）

番地	メモリの内容	
	:	main にとっての
26	7	···a
27	3	···b
	:	
	:	hoge にとっての
123		···x
124		···y
	:	

● hoge02.c の場合

7,9 行目 hoge01.c と同じ

10,16 行目 &b は,「b **の番地**」を表す．main は，a の**値**「7」と b の**番地**「27 番地」を hoge に渡す．hoge はそれらを自分で用意した場所に格納し，x,y と呼ぶことにする.

18 行目 *y は,「y **が指す場所の値**」を表す．今の場合 y には「27 番地」が格納されているので，*y は 27 番地の値を表す.「x の値」と「y が指す場所の値」を足して「y が指す場所の値」とするので，27 番地が書きかわる.

19 行目 x と*y の値を出力（7 と 10 が表示される）

11 行目 hoge をぬけて main に戻ってきた．b の値は hoge の実行中に書きかわっている．それらを出力（7 と 10 が表示される）

番地	メモリの内容	
	:	main にとっての
26	7	···a
27	3	···b
	:	
	:	hoge にとっての
123		···x
124		···y
	:	

Q8. 次のような仕様の関数 f1() の定義を書きなさい．

- 関数名は f1，戻り値なし
- 引数は３つ（仮引数の名前を順に x，y，z とする）で，それぞれ，double 型か double 型へのポインタ型のどちらか
- この関数は，「x と y の和を z に代入する」処理を行う．下記左のプログラムの一部に対して右のような実行結果が得られるものとする．

```
double x;
f1( 1.5, 2.0, &x );
printf("x = %f\n", x);
```

［実行結果］

```
x = 3.5
```

［発展］ 関数の引数にポインタを用いる必要のある場合とない場合の違いがわかれば，printf() と scanf() の引数の指定の仕方が違う理由がわかりますね．

```
int x;
scanf("%d", &x);
printf("%d", x);
```

Q9. 以下のアイウエは，右の f01.c から f04.c までのプログラムの「関数の定義」と１対１に対応している．最も適切な対応づけはどのようになるか答えなさい．また，それぞれのプログラムを実行するとどのような出力が得られるか答えなさい．

ア

```
void f(int s, double t)
{
  printf("s = %d, t = %f, s + t = %f\n",
         s, t, s - t);
}
```

イ

```
double f(int x, int y)
{
  double z = (double)x / y;
  return z;
}
```

ウ

```
void f(int a, int b)
{
  printf("a = %d, b = %d, a + b = %d\n",
         a, b, a + b);
}
```

エ

```
void f(int x, double *y)
{
  *y = x * 0.1;
}
```

f01.c

```
 1  #include <stdio.h>
 2
 3  「プロトタイプ宣言」
 4
 5  int main(void)
 6  {
 7    int x = -1;
 8    f(7, x);
 9  }
10
 :  「関数の定義」
```

f02.c

```
 :
 5  int main(void)
 6  {
 7    int a = 3;
 8    f(a, 3.1415);
 9  }
10
 :  「関数の定義」
```

f03.c

```
 :
 5  int main(void)
 6  {
 7    double x;  int y = 10;
 8    x = f(y, 3);
 9    printf("x = %f\n", x);
10  }
11
 :  「関数の定義」
```

f04.c

```
 :
 5  int main(void)
 6  {
 7    int a = 3; double b;
 8    f(a, &b);
 9    printf("b = %f\n", b);
10  }
11
 :  「関数の定義」
```

★ 1.3 配列

Q10. 以下の array01.c を実行するとどんな出力が得られるか.

Q11. 右の array02.c は，関数を用いて array01.c と同じ出力が得られるようにしようとしたものである. 16 行目の関数呼び出しを参考にして，プロトタイプ宣言（3 行目）と関数の定義（23 行目以降）を書きなさい.

array01.c

```
 1  #include <stdio.h>
 2
 3  int main(void)
 4  {
 5    int a[10];
 6    int n = 4, i;
 7
 8    for(i = 0; i < n; i++){
 9      a[i] = 2*i + 1;
10      printf("%d %d\n", i, a[i]);
11    }
12    printf("-----\n");
13
14    for(i = 0; i < n; i++){
15      a[i] *= 3;
16    }
17
18    for(i = 0; i < n; i++){
19      printf("%d %d\n", i, a[i]);
20    }
21
22    return 0;
23  }
```

Q12. int 型などの普通の変数と違い，array02.c のように配列を関数に渡した場合，関数中で値を変えたら main の側でも値が変わります. どうしてそういうことになるのでしょう？

array02.c

```
 1  #include <stdio.h>
 2
 3
 4
 5  int main(void)
 6  {
 7    int a[10];
 8    int n = 4, i;
 9
10    for(i = 0; i < n; i++){
11      a[i] = 2*i + 1;
12      printf("%d %d\n", i, a[i]);
13    }
14    printf("-----\n");
15
16    func(a, n);
17
18    for(i = 0; i < n; i++){
19      printf("%d %d\n", i, a[i]);
20    }
21
22    return 0;
23  }
24
25
26
27
28
29
30
31
32
```