ゼミC　第1回

8月4日

C言語の復習・C++の基礎

エレクトロニクス研究部

C言語の復習

- 何をするの?

主にゼミAでやったことの復習です

- printf/scanf
- If文/for文/while文
- 配列
- 構造体/関数
- ポインタ

printf

```
printf("Hello, world");
```

printf関数
C言語で文字列を表示するための関数。

- 出力時表示されない文字列を使う。これをエスケープシーケンスという。

¥n　改行。

¥t　tabを挿入する。

¥a　警告音を出す。　…など

改行をする際に使う「¥」やその他「?」や「”」なども特殊文字に辺り普通には出力されない場合がある。この文字を出力したい際はその文字の前に「¥」を置く。

- 例

入力

```
printf("Hello¥¥nworld");
```

出力

```
Hello¥world
```

- 出力する際””に囲まれた文字列を文字列リテラルという。
- 数値を出力する際には「%d」などを使うがこれを出力変換指定子という。

出力変換指定子	引数の型
%d	int型の実数　10進数
%f	double型の実数
%c	char型の文字
%s	char型の文字列

「%」を出力したい場合は「%%」と入力する。

scanf

```
scanf(“%d“,&a);
```

scanf関数

キーボードから入力する際に使う。入力には入力変換指定子を使う。入力変換指定子は上の文だと「%d」にあたる。

入力変換指定子	引数の型
%c	char型の文字
%s	char型の文字列
%d	int型実数
%lf	double型実数 （出力と違うので注意）

練習問題その1

double型変数を二つ読み取り、その二つの値を足して3で割った値を出力するようにせよ。

解答1

```
#include <stdio.h>
int main(void){
        double a,b;
        scanf("%lf",&a);
        scanf("%lf",&b);
        printf("(a+b)/3=%f¥n",(a+b)/3);
        return 0;
}
```

if文

- 条件判断をする際はif文を使う。

```
if(条件文) 文 ;
```

条件文が真(≠0)の時、文を実行。

偽(＝0)の時は実行しない。

- 条件文にて値を比較する際、演算子を使うことがある。

演算子	真の時	偽の時
a==b	aとbが等しい	aとbが等しくない
a!=b	aとbが等しくない	aとbが等しい
a>b	bよりaのほうが大きい	aよりbのほうが大きい
a<b	aよりbのほうが大きい	bよりaのほうが大きい
a>=b	aがb以上	bがa以上
a<=b	bがa以上	aがb以上

また、ひとつのif文で複数の条件文を比較する際も演算子を使う。それを論理演算子という。

演算子	真の時	偽の時
A&&B	AND、AとB両方の文が成立する	NAND（ANDの逆）、どちらか一方でも成立しない
A||B	OR、AとBどちらか一方でも成立する	NOR、AとB両方どちらも成立しない
!A	NOT、Aが成立しない	Aが成立する

- もし条件文を実行しなかった際に限って実行したいことがある際はelse文を使う。

```
if(条件文) 文
else 文　;
```

- 例

```
if(a==1){
        printf(“aは1です¥n”)
}
else{
        printf(“aは1じゃないです¥n”)
}
```

aの値が1じゃない時二つ目の実行文を実行

- 条件が成り立たない時の条件分岐を増やしたいときは else if文を使用する。

```
if(条件文) 文
else if(条件文) 文　;
```

- 例

```
if(a==10){
        printf(“aは10です¥n”)
}
else if (a<10){
        printf(“aは10以下です¥n”)
}
else{
        printf(“aは10以上です¥n”)
}
```

練習問題2

- 二つの値を読み取る。その後、aがbより大きければ「aのほうが大きい」と出力、bが大きければ「bのほうが大きい」と出力、同じでaとbがどちらも正の数「aとbは同じ正の数」、どちらも同じで負の数あるいは0なら「aとbは負の数」と出力せよ。

解答2

```
#include<stdio.h>
int main(void){
        int a,b;
        scanf("%d",&a);
        scanf("%d",&b);
        if(a>b){
                printf("aがでかい¥n",a);
        }
        else if(a<b){
                printf("bがでかい¥n");
        }
        else if(a==b&&a>0){
                printf(" aとbは同じ正の数¥n");
        }
        else{
                printf(" aとbは負の数¥n");
        }
        return 0;
}
```

for文

for(変数の初期化;条件文; 変数の処理) 文

処理を繰り返したいときに使うのがfor文。繰り返す回数が決まってる時によく使う。

例えば変数の初期化で「i＝0」、条件文で「i＞10」、変数の処理で「i＋＋」と置くと、文を10回行う。

iの初期値を0とし、文を一回実行することにi＋＋、iの値を一つ増やす。そしてこれをi＞10になるまで実行する。

文の中でiの値を増減することもできる。

while文

```
while(条件文) 文 ;
```

条件文の中が等しい限り、文を繰り返し実行し続けるループ文。

例えば条件文にi＞10とおくと、iが10である限り文を実行し続ける。iの値の変更は文の中で行う。

また条件文に1とおくと無限ループ、文を無限に実行し続ける。while文を使う際は無限ループが多い。

文の中にbreak;と置くとループ文を抜ける。if文と組み合わせるといい感じ

do~while文

基本的にはwhile文と同じ感じ

```
do{ 文; } while(条件文) ;
```

条件文が真である限り文を実行し続ける。while文と違って実行後に条件文の判定に入る。

あんまり使わない

switch文

- 文を多岐の条件にわたって実行したい際にはswitch文を使う。else文と同じといえば同じ。

```
while(a) {
        case 1:printf(“aは1¥n”); break;
        case 2:printf(“aは2¥n”); break;
        default : printf(“それ以外”); break;
}
```

aの値が1の時、2の時、それ以外で実行する文が違う。

一応else if(a==2){　～とでもできるが条件文が多く面倒な時はこれを使う。breakを使わないと延々とcaseを比較し始めるので忘れないように注意。

break文　continue文

break文はwhile文で軽く説明したものと同じ。ループ文を抜ける。
continue文は同じくループ文にて使う。continue文以下の処理をスキップし、次のループに移りたい際に使用する。ちなみにループ文はfor、while、do~while文など。

```
while(a!=10){
        i++;
        if(a==1)break;
        if(a==5)continue;
        printf("ループ文実行¥n");
}
```

aが1のときループ文を抜ける。
aが5のとき文の出力をぜず、次にループに移る。

練習問題3

値を10回読み込み、その値を5倍した数を出力するプログラムを作れ。ただし、マイナスの値が入力された時はスキップし、0が入力された時は終了させること。

解答3

```
#include<stdio.h>
int main(void){
        int i,a;
        for(i=0;i<10;i++){
                scanf("%d",&a);
                if(a<0){continue;}
                if(a==0){break;}
                printf("%d¥n",a*5);
        }
        return 0;
}
```

配列

- 複数の変数を一気に宣言したいときは配列を使おう

```
int a[3];
```

int型の整数a[0],a[1],a[2]が宣言された。a[3]は宣言されていないことに注意。

```
int i=3;
int b[i];
```

上の宣言の仕方でも可能。b[0]~b[2]が宣言される。

- 配列は宣言時にまとめて初期化を行うこともできる

```
int a[5]={0};
```

こうすることでa[0]~a[4]に0が代入される。

```
int a[5]={2};
```

a[0]に2が代入される。それ以外a[1]~a[4]は0

```
int a[5]={2,3,4};
```

これだとa[0]に2,a[1]に3、a[2]に4が代入、それ以外は0

ちなみに配列の個数の数を要素数（a[100]だと100のこと）というがこれは宣言時に省略できる。

```
int a[]={0,1,2,3,4,5,6,7};
```

これはa[8]を宣言したことになり、初期化はa[0]は0,a[1]は1、～a[7]は7の通りになる。

ちなみに配列の中の値の初期化は宣言時にしかできない。宣言時以外、文の途中でa[10]={0};とやってもエラーと出るので注意。

宣言時要素数に変数が使える、と説明したがそれ以外でももちろん変数は使用可能。

```
i=9;
printf(“%d¥n”,a[i]);
```

こうすることでa[9]の値が出力される。

2次元配列というものも存在する。

```
int a[5][5];
```

こうすることで
a[0][0],a[1][0],a[2][0],~,a[0][1],a[0][2],~~~~
ってかんじで変数が25個宣言される。
ちなみにa[5][5][5]ってかんじで3次元配列、
a[5][5][5][5]って感じで4次元配列～と宣言もできる。

練習問題4

50以下の整数を一つ入力する。次にその入力した整数だけ再び整数を入力する。そして入力した整数すべてと、入力した整数群の和を出力せよ。

- 一例

入力：3,5,6,7

出力：

5,6,7が入力されました。和は18です。

解答4

```
#include<stdio.h>
int main(void){
		int n,i,sum=0;
		int a[50]={0};
		scanf("%d",&n);
		if(n>50){
				printf("数が大きすぎます！¥n");
				return 0;
		}
		for(i=0;i<n;i++){
				scanf("%d",&a[i]);
				sum=sum+a[i];
		}
		for(i=0;i<n;i++){
				if(i==0){
						printf("%d",a[0]);
				}
				else{
						printf(",%d",a[i]);
				}
		}
		printf("が入力されました。和は%dです。¥n",sum);
		return 0;
}
```

構造体

```
struct Student{
        int number;
        int point;
};
```

セミコロンを忘れないように注意。

これでstruct Student a;と宣言すると、

a.number、a.pointにアクセスできる。複数のデータを扱うときに便利。

C++、ゲーム作りにはこの構造体の発展であるクラスを使用。

練習問題5

生徒のNoと数学、英語の点数と平均を格納する構造体を作り、生徒3人の数学、英語の点数を入力して、各生徒のNo、数学、英語の点数と平均点を出力せよ。

```
struct Student{
        int no;
        int math;
        int english;
        int average;
};
int main(void){
        struct Student a[3];
        int i;
        for(i=0;i<3;i++){
                a[i].no=i;
                scanf("%d%d",&a[i].math,&a[i].english);
                a[i].average =(a[i].math+a[i].english)/2;
        }
        for(i=0;i<3;i++){
                printf("no:%d 数学:%d 英語:%d 平均%d ¥n",a[i].no,a[i].math, a[i].english,a[i].
average);
        }
        return 0;
}
```

関数

同じプログラムを何回も繰り返すときはその同じプログラムを何回も書くよりかは関数を作成、それを何度も呼び出すのがよい。
以前に作成したプログラムを再び使うことを再利用という。

- 例　渡された変数を出力する

```
void print(int a){
        printf("%d",a);
}
int main(void){
        int i=10;
        print(i);
        return 0;
}
```

- 注意点としてプログラム内で使用する関数はmain文前でプロトタイプ宣言する、あるいは関数自体を作成しないとmain文で呼び出した際、正常に呼び出すことができない。

```
void print(void);
```

これは前の例のプログラム内で使用した関数のプロトタイプ宣言。

前の関数printではmain関数から変数を受け取り、その変数を出力していた。このように関数に渡す変数を引数という。関数内で引数を使いたい際、ここではint型aとdouble型bを使いたいとすると

```
void hello(int a,double b){
        文
}
int main(void){
        文
        hello(a,b);
}
```

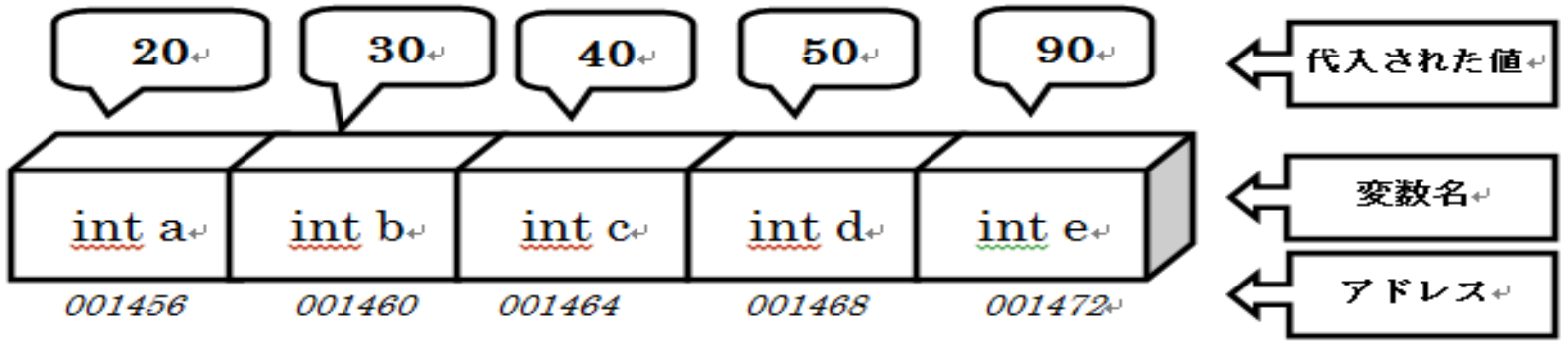
とする。引数が複数ある際は上のように「,」で区切る。

関数には型がある。

int型だと戻り値が必要。returnってやつ
void型だと戻り値はいらない。
main文はint型。

ポインタ

変数がしまわれている場所を参照できる。



宣言の方法は変数の形式 *変数名
例）int *a;
普通に変数に&を入れることでアドレスを参照できる。
例）int a;　　p=&a;
これで変数pにaのアドレスが代入された。

```
#include <stdio.h>
int main(void)
{
	int *a,b;
	a = &b;
	printf("a = %d¥n",a);
	printf("&b = %d¥n",&b);
	return 0;
}
```

上のプログラムではポインタ変数a、普通の変数bを宣言。その後aにbのポインタを代入している。

出力される値は同じになっている。

- ポインタのまとめ

変数	値	アドレス
int a; (int型の変数a)	a	&a
int *p; (int型のポインタp)	*p	p

普通の変数と同じくポインタにも宣言するとその中にはデタラメな値が代入されている。ポインタを代入したい際はNULLを代入するとよい。これをヌルポインタという。

ちなみにポインタ自身にもポインタが存在する。
printf(“%d”,&p)で出力できる。
ポインタのポインタも宣言でき、その場合は
int **p;　とする。

いままで使用してきた配列も実はポインタである。

char a[5]という配列を宣言した際は実はchar aを宣言、そっからポインタの続いた5つの値を連続で宣言したということになる。

そのため&a[0]とaは等しく、&a[1]とa+1は等しい。

ポインタ		アドレス(例)
a	&a[0]	1000
a+1	&a[1]	1001
a+2	&a[2]	1002
a+3	&a[3]	1003

練習問題6

- 2つのポインタ変数を受け取り、それぞれの値を2倍にした上入れ替える関数void dswap(int *a,int *b)を作れ。

解答6

```
void dswap(int *a,int *b){
        int sw=*a;
        *a=*b;
        *b=sw;
        *a*=2;
        *b*=2;
}
```

C言語の復習は以上です！
次はC++についてやっていきます！