

第三回C言語講習

関数～配列、文字列

目次

- 関数
 - 関数の定義
 - 引数・返り値
 - プロトタイプ宣言
 - 再帰関数
- 文字
- 配列
 - 配列の宣言
 - 配列を使う
- 文字列
- 多次元配列
- 配列、文字列の諸関数

目次

- 関数 @
 - 関数の定義
 - 引数・返り値
 - プロトタイプ宣言
 - 再帰関数
- 文字
- 配列
 - 配列の宣言
 - 配列を使う
- 文字列
- 多次元配列
- 配列、文字列の諸関数

関数

数学の関数の話

f(x)

$$f(x) = ax^2 + bx + c$$

これは
 ax^2+bx+c
という計算の手続きを

$f(x)$

とまとめて表記している

プログラムの話

例えば...

プログラムの中で
「二つの変数があって
そのうち大きい方を2倍して
小さいほうを3倍した数」

が必要な箇所がたくさんあったとする。

例えば...

int a,b;と宣言されてるとして

```
if(a<b){  
    a*3+b*2;  
}else{  
    a*2+b*3;  
}
```

みたいな処理がたくさん

問題点

何回も書いてるとミスタイプするかも？

もしもその処理を

ちょっと変更したって時に

全部の箇所を書き換えなくちゃならない

プログラムがやたらと長くなりがち

同じ処理をまとめよう

そこで関数

関数にも宣言が必要

C言語の関数宣言

```
戻り値の型 関数の名前(引数の型 引数の仮の名前 ...){  
    関数の処理の内容;  
    return 戻り値の値;  
}
```

Hello World(第一回C言語講習より)

```
//hello.c
```

```
#include<stdio.h>
```

```
int main(void){
```

```
    printf("Hello World\n");
```

```
    return 0;
```

```
}
```

今まで
main関数を定義していた

関数の諸々の値

C言語の関数宣言

```
戻り値の型 関数の名前(引数の型 引数の仮の名前 ...){  
    関数の処理の内容;  
    return 戻り値の値;  
}
```

引数

$f(x)$ における x

関数に与える値

引数

```
int function(int parameter, int parameter2){  
    return (parameter+parameter2)/2;  
}
```

parameterという名前の変数として使える。
(スコープはこの関数の中だけ)

例えば

```
meal hamburg_maker(food meat,food onion,food egg){  
    meal hamburg;  
    hamburg = cook(meat,onion,egg);  
    return hamburg;  
}
```

この関数は
food型の引数を3つ取り
meal型の値を返す

もう少し現実的な例

```
int add_num(int x, int y){  
    int ans;  
    ans = x + y ;  
    return ans;  
}
```

int 型の引数を二つ取り
その和を返す

C言語の関数宣言

```
戻り値の型 関数の名前(引数の型 引数の仮の名前 ...){  
    関数の処理の内容;  
    return 戻り値の値;  
}
```

返回值

f(x)の値

$$y=f(x)$$

$f(x)=5$ だとすると

$$y=f(x)$$

$$y=5$$

これを
 $f(x)$ が5を返した
と表現する

値の返し方

return 文

return 返す値;

```
return 0;
```

これは0を返している

return 文

return 文が実行されると

その時点で関数は値を返して終了する。

つまりreturn 文以降にかかれた文は

実行されない。

ちなみに
大抵のプログラミング
において

関数は一つしか
値を返せない

实例

実際に関数を使ってみる

void 型

引数や返り値が不要ない

`int main()`..のように省略
してもいいけど..

**明示的に
無いことを示したい**

そこでvoid型

引数を持たない関数

```
int none_parameter(void ){  
    hogehoge;  
    return 0;  
}
```

引数にvoidが指定されている。

返り値を持たない関数

```
void none_value(int a){  
    fugafuga;  
    return ;  
}
```

返り値にvoidが指定されている。

自作した関数を
自作した関数で使う

使う関数よりも
前に使われる関数
が宣言されていればOK

实例

main関数が下の方に...

関数のプロトタイプ宣言

**main関数より前に
宣言するのは
関数の概要だけでいい**

**関数の本体はmain関数の
後に書く**

プロトタイプ宣言

返り値の型 関数名(引数の型 引数の仮の名前);

!!最後の;を忘れないように!!

プロトタイプ宣言(本体)

プロトタイプ宣言をしたあと
main関数の後に

```
戻り値の型 関数名(引数の型 引数の仮の名前){  
    処理の内容;  
    return 返す値;  
}
```

普通の関数宣言と一緒にのものを書いただけ

实例

目次

- 関数
 - 関数の定義
 - 引数・返り値
 - プロトタイプ宣言
 - 再帰関数@
- 文字
- 配列
 - 配列の宣言
 - 配列を使う
- 文字列
- 多次元配列
- 配列、文字列の諸関数

再帰関数

超超超發展的內容

そんなんがあるんだ

へー

程度でOK

再帰関数

**自分の定義の中に
自分自身を含む関数**

よくある再起関数の例

```
int factorial(int x){  
    if(x==0){  
        return 1;  
    }else{  
        return x*factorial(x-1);  
    }  
}
```

xの階乗(x!)を求める関数

階乗 $x!$ の定義

階乗の定義

$f(x)=x!$ とおくと

$$f(x) = 1 \quad : (x=0)$$

$$= x * f(x-1) \quad : (\text{otherwise})$$

よくある再起関数の例

```
int factorial(int x){  
    if(x==0){  
        return 1;  
    }else{  
        return x*factorial(x-1);  
    }  
}
```


このように一部の処理は
再帰関数を使うと
簡潔にかける

目次

- 関数
 - 関数の定義
 - 引数・返り値
 - プロトタイプ宣言
 - 再帰関数
- 文字@
- 配列
 - 配列の宣言
 - 配列を使う
- 文字列
- 多次元配列
- 配列、文字列の諸関数

文字

コンピューターは
数値しか扱えない

文字を表現するには
どうするか

昔の人は考えた

文字と数に対応付けよう

'A'にはこの数字
'B'にはこの数字

⋮

これがASCIIコード

残念なことに
全角文字は
ASCIIコードには
含まれていません

C言語における文字

C言語における文字

—文字ずつchar 型の値を持ちます。

(char = character の略)

C言語における文字

一文字を

「'」シングルクォーテーション

で囲む

C言語における文字

囲むと

その文字を表す

ASCIIコードの数値を意味する

ASCIIコードにおいて

'A'=65
なので


```
char mozi;  
mozi = 'A';  
printf("%d",mozi);
```

こうすると

65

と出力される

文字を意味する
書式指定子

`%c`

を使うと

```
char mozi;  
mozi = 'A';  
printf("%c",mozi);
```

こうすると

A

と出力される

直接数値をいれてもいい

```
char mozi;  
mozi = 65;  
printf("%c", mozi);
```

こうしても

A

と出力される

ちなみにA,B,C,,
と連番になっているので

```
char mozi;  
mozi = 'A';  
printf("%c",mozi+1);
```

こうすると

B

と出力される

実演

講習長くね？

目次

- 関数
 - 関数の定義
 - 引数・返り値
 - プロトタイプ宣言
 - 再帰関数
- 文字
- 配列 @
 - 配列の宣言
 - 配列を使う
- 文字列
- 多次元配列
- 配列、文字列の諸関数

配列

たとえば
学生の成績のデータ

学生の成績データ

ヤマダさんは50点

タナカさんは60点

スズキさんは70点

プログラム
ではどうする？

学生の成績データ

```
int yamada,tanaka,suzuki;  
yamada = 50;  
tanaka = 60;  
suzuki = 70;
```

これでもいいけど...

学生の数が増えたとき大変

データをまとめたい

そこで配列

配列

同じ型のデータをまとめて

番号を付けたもの

具体的には

こんなかんじ

0	1	2	3	4	5	...		
---	---	---	---	---	---	-----	--	--

C言語では

型 名前[欲しい箱の数];

```
int a[10];
```

`int a[10];`と宣言すると

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

番号が0から9なのに注意

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

配列の番号は0から始まる

配列を使う

普通の変数と同じ

先ほどの例で
3番目の箱
に値を代入したい時
(例えば10)

```
a[3]=10;
```

ではなくて

```
a[2]=10;
```

```
a[2]=10;
```

最初から順に0,1,2..と代入

配列に代入

```
int a[10];
```

```
a[0] = 0;
```

```
a[1] = 1;
```

```
a[2] = 2;
```

配列に代入 (for文を使って)

```
int a[10];
```

```
int i;
```

```
for( i=0;i<3;i++){
```

```
    a[i] = i;
```

```
}
```

配列に代入 (for文を使って)

```
int a[10];
```

```
int i;
```

```
for( i=0;i<3;i++){
```

```
    a[i] = i; //何番目かを変数で指定
```

```
}
```

ただし
宣言するときに箱の数を**変数**
では指定できない

`int h = 10; int a[h];`
のような書き方は無理

実演

初期化

変数を宣言と同時に代入
するときだけ
特別な書き方


```
int a[10] = {0,1,2};
```

{ で囲んで

, で区切る

変数の宣言と同時に代入する
ることを
初期化
と言います

配列にまつわる用語

配列の箱の数:要素数

配列の箱それぞれ:要素

何番目かを指す数字:**添字**

配列の用語(例)

```
int a[10];
```

```
//要素数10の配列を宣言している。
```

```
a[4]=5;
```

```
//アクセスしている添字は4
```

```
//これは頭から数えて5番目
```

配列に関する**注意点**

配列の長さがわからない

`int a[10];`
って宣言している時

`a[1000] = 10;`
と書いても
コンパイラはスルー

この時どうなるかは
誰にもわからない

**配列の長さを別の変数で保
持しておく等
工夫が必要**

目次

- 関数
 - 関数の定義
 - 引数・返り値
 - プロトタイプ宣言
 - 再帰関数
- 文字
- 配列
 - 配列の宣言
 - 配列を使う
- 文字列 @
- 多次元配列
- 配列、文字列の諸関数

文字列

これでやっと文が扱える

文字列

文字の配列

char 型の配列

ただ

文の終わりに特殊な文字が
入っている

NULL文字 \0

これで文の終わりを示す

C言語では '\0'

“Hello”という文字列は

こんなかんじ

'H'	'e'	'l'	'l'	'o'	'\0'
-----	-----	-----	-----	-----	------	-----	-----	-----

多次元配列

配列は
2次元や3次元にもなる

```
int mat[5][3];
```

アクセスも
普通の配列と一緒に

```
mat[0][1]= 10;
```

文字列、多次元配列 それぞれ実演

便利な関数も
それぞれ実演

おわり

それでは
各自演習に入ってください