

プログラミング言語基礎 [その6]

今日の目標！

http://www.ipc.fukushima-u.ac.jp/~p058

福島県立医科大学 物理 吉田 宏
E-Mail: yoshidah@fmu.ac.jp

2004/5/25

プログラミング言語基礎(その6)

1

2004/5/25

プログラミング言語基礎(その6)

2

2. 変数(データ型)

型	データ	バイト幅	範囲
char	文字	1byte(8bit)	文字 or -128~127の整数
int	(符号付)整数	4byte(32bit)	$-2^{31} \sim 2^{31}-1$ の整数
float	単精度実数	4byte(32bit)	約 $10^{-38} \sim 10^{38}$ (7桁有効)
double	倍精度実数	8byte(64bit)	約 $10^{-308} \sim 10^{308}$ (15桁有効)

データのメモリの確保の仕方

1bit = 0か1の2通り 1byte=8bit = 2^8 通り = 256 通り
データ型ごとに使うメモリの大きさが異なる
効率よくメモリーを使うことができる

2004/5/25

プログラミング言語基礎(その6)

3

printfとscanf

変数型	宣言文	printfの中身	scanfの中身
整数	int i;	printf("%d\n", i);	scanf("%d", &i);
文字	char c;	printf("%c\n", c);	scanf("%c", &c);
文字列	char a[10];	printf("%s\n", a);	scanf("%s", a);
単精度実数	float x;	printf("%f\n", x);	scanf("%f", &x);
倍精度実数	double y;	printf("%f\n", y);	scanf("%lf", &y);

データ型ごとに出力・入力の書式の指定が異なる

2004/5/25

プログラミング言語基礎(その6)

4

```

1:/* ex0201.c y=x*x+2*x+1を計算するプログラム */
2:/* 作成日 2004/5/19 */
3:/* 作成者 xxxxxx */
4:#include <stdio.h>
5:
6:/* 開始 */
7:int main()
8:{
9:    int x,y; /*x,yを整数型変数として宣言。(宣言文)*/
10:
11:    printf("xを入力してください");
12:    scanf("%d",&x); /* xに値を整数型のデータとして入力*/
13:    float x,y; /*課題2.3*/
14:    y=x*x+2*x+1; /*scanf("%f",&x);(課題2.4)*/
15:
16:    printf("xが%dのとき, y=x*x+2*x+1は %d です\n",x,y);
17:    return 0;
18:}
19:/* 終了 */
    
```

2004/5/25

プログラミング言語基礎(その6)

5

課題2.3 ex0202.c

ex0201.cの9行目

int x,y; → float x,y;

課題2.4 ex0203.c

ex0202.cの12行目

scanf("%d",&x); → scanf("%f",&x);

課題2.5 ex0204.c

ex0203.cの16行目

printf("xが%dの... → printf("xが%fの...)

正しくない結果

正しい結果

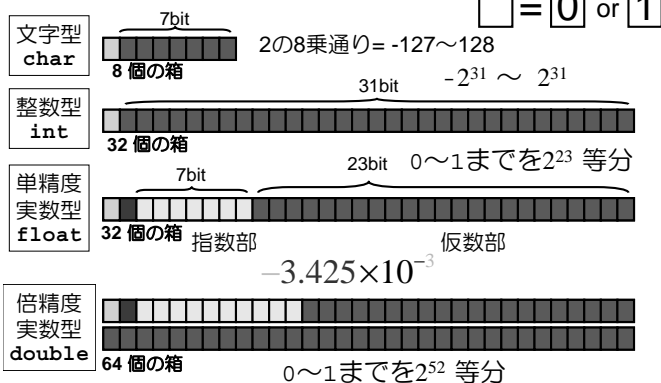
2004/5/25

プログラミング言語基礎(その6)

6

各変数型のメモリの確保

□ = 0 or 1

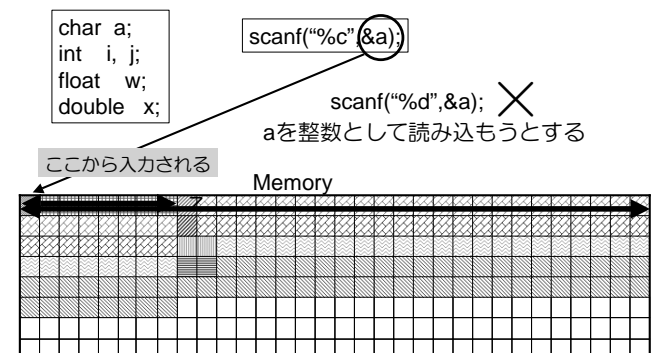


2004/5/25

プログラミング言語基礎(その6)

7

コンピュータ内のメモリの確保



2004/5/25

プログラミング言語基礎(その6)

8

「°」を「radian」に変換するプログラム

```

1: /* ex0206.c 角度の換算 degree -> radian */
2: /* 作成者 xxxxx */
3: /* 作成日 2004/5/26 */
4:
5: #include <stdio.h> /* stdio.hをインクルードする */
6: int main() /* main関数 */
7: {
8:     int radian, degree=0; /* 整数型radian, degreeの宣言 */
9:     float PI=3.141592654; /* PI 円周率 */
10:
11:     while(degree <= 180){
12:         radian = degree*PI/180; /* ラジアン換算 */
13:         printf("%d %d\n", degree, radian); /* 結果の出力 */
14:         degree+=10; /* degreeを10増やす */
15:     }
16:     return 0; /* 終了 */
17: }
    
```

Try 課題 2.6~2.8

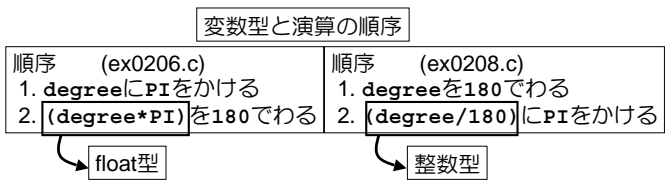
ex0206.cとex0208.cの違い

```

ex0206.cの12行目
    degree*PI/180
ex0208.cの12行目
    degree/180*PI
    
```

```

宣言文
int degree, radian;
float PI=3.141592654;
    
```



変数型と演算

aob	int	float	double
int	int	float	double
float	float	float	double
double	double	double	double

整数型の変数に実数を代入すると 小数点以下は切り捨てられる

```

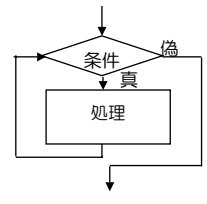
int a;
a=3.123;
printf("%d\n", a);
printf("%f\n", a);
    
```

Try 課題 2.9

while文

```

while(条件){
    処理1;
    処理2;
    処理3;
    :
}
    
```



```

int n=0;
while(n<=5){
    printf("-----%d\n", n);
    ++n;
}
    
```

条件の書き方

while文, for文, do while文, if文などで次のような条件判断が使われる

意味	表現	使用例
$a > b$	aはbより大きい	$a > b$ while (a>b) {...}
$a < b$	aはbより小さい	$a < b$ while (a<b) {...}
$a \geq b$	aはb以上	$a \geq b$ while (a>=b) {...}
$a \leq b$	aはb以下	$a \leq b$ while (a<=b) {...}
$a = b$	aとbは等しい	$a == b$ while (a==b) {...}
$a \neq b$	aとbは等しくない	$a != b$ while (a!=b) {...}

2. 配列 同じ変数型の集まり ベクトルや行列など

44人の「プログラミング言語基礎」の期末テストの平均値 配列を使わないと

```

int a01,a02,a03,a04,a05,a06,a07,a08,a09,a10,a11;
int a12,a13,a14,a15,a16,a17,a18,a19,a20,a21,a22;
int a23,a24,a25,a26,a27,a28,a29,a30,a31,a32,a33;
int a34,a35,a36,a37,a38,a39,a40,a41,a42,a43,a44;
int heikin,total;
    
```

```

total=a01+a02+a03+a04+a05+a06+a07+a08+a09+a10+ a11
+a12+a13+a14+a15+a16+a17+a18+a19+a20+a21+a22
+a23+a24+a25+a26+a27+a28+a29+a30+a31+a32+a33
+a34+a35+a36+a37+a38+a39+a40+a41+a42+a43+a44;
heikin=total/44.0;
    
```

配列を使うと

```

/* 変数を宣言するとき */
int heikin, total, i, a[44];
/* 合計&平均を求める */
i=total=0;
while(i<44){
    total+=a[i];
    i++;
}
heikin=total/44.0;
    
```

→ a[440]; (初期化)
→ iが44になったら、演算を止める
→ totalにa[i]のデータを加算する
→ iを1増やす
→ total/410.0;

44人->440人とするには...

配列の宣言の仕方

```

int a[10]; /* 整数型の配列aとして要素数10の変数を宣言 */
float a; /* 同じ関数内で同名の変数は宣言できない */
    
```

この宣言で a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9] の整数型変数が使用できる。

- 配列の大きさ N (=10) を宣言文で指定する
- 0番目からN-1番目まで(a[0],...,a[9]) が変数として使える
- 配列として宣言した変数は、同じ関数内で別の変数の名前と重複してはいけない

Try 課題 2.10&11