

## 第3回簡単なデータの入出力



1

## 今回の目標

- 変数を理解する。
- 変数や定数の型を理解する。
- 入出力の方法を理解する。

☆入出力のあるプログラムを作成する。

2

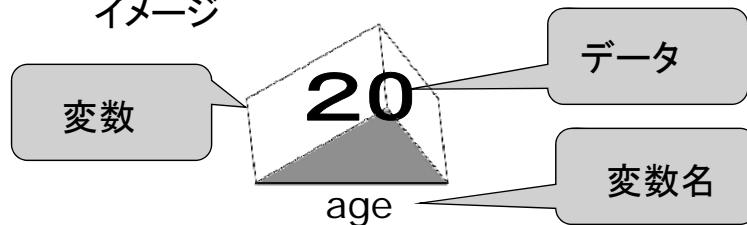
## 変数

変数: データを入れる入れ物。

変数名: 変数の名前。

規則にのっとった文字列で命名する。

イメージ



3

## 数学とC言語の変数の違い

### 数学

入れ  
られる  
データ

主に数で、  
整数、実数  
の区別をしない。

### C言語

主に数、文字で、  
種類毎に区別する。  
特に、整数と実数も区別する。  
(型という考え方方が生じる)



変数名

慣用的に決まっており、  
x, y, t等の1文字  
が多い。

長い名前を自分で命名  
できる。



4

## 命名規則

[規則] 名前(変数名、関数名等)は英字、数字あるいは  
\_\_(アンダースコア)だけからなり先頭は数字以外の文字である。

(スタイル規則B参照)

変数例

age	x_coordinate
i	y_coordinate
j	x1
k	y1

なるべく意味のある文字列にする事。

main内で宣言する変数は英小文字と数字  
\_\_(アンダースコア)だけを用い英大文字は用いない事。

5

## 予約語(キーワード)

予約語

auto	break	case	char
continue	default	do	double
else	for	goto	if
int	long	register	return
short	sizeof	static	struct
switch	typedef	union	unsigned
void	while		

C言語の予約語は以上である。

なお、printfとかscanfとかは、ライブラリ中で定義されている。

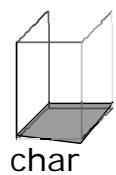
注意: 変数名や関数名に予約語を用いてはいけない。  
(予約語以外で、命名する。)

6

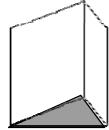
## C言語の代表的なデータ型

データは、種類ごとに異なる扱いをしなければならない。  
種類は、型として区別される。

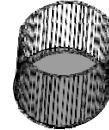
char	1バイトの整数型(1つの文字を表す型)
int	整数型
float	単精度浮動小数点型
double	倍精度浮動小数点型



char



int



double

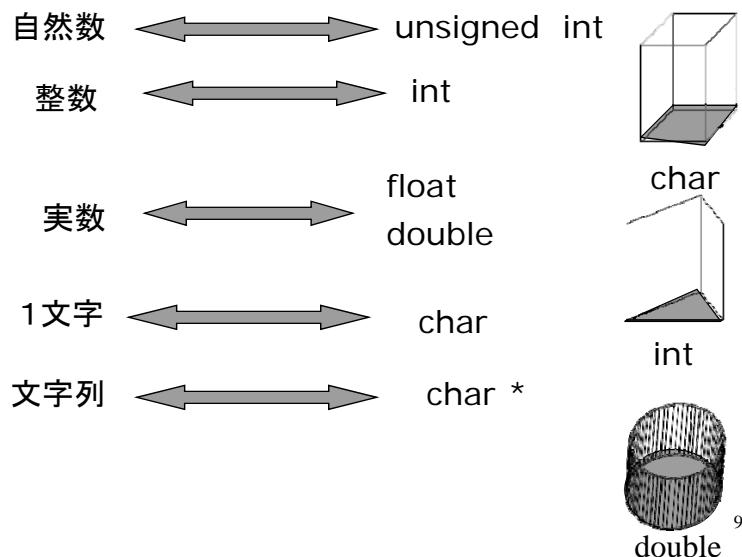
7

## C言語のデータ型が扱える範囲

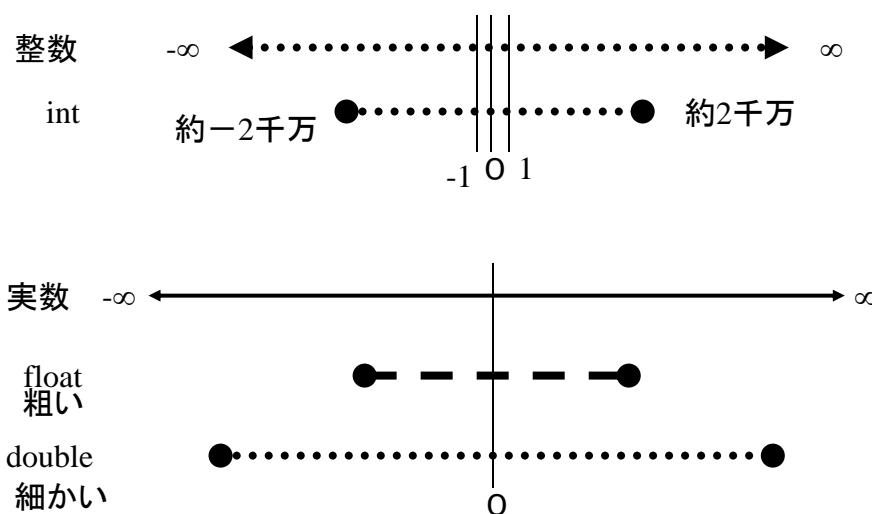
型	バイト長	表現範囲
char	1	0~255
int	4	-214748648~21483647
short int	2	-32768~32767
unsigned int	4	0~4294967295
float	4	$\pm 1.0 \times 10^{-37}$ ~ $\pm 1.0 \times 10^{38}$
double	8	$\pm 1.0 \times 10^{-307}$ ~ $\pm 1.0 \times 10^{308}$

8

## 数学の概念とC言語の型



## 数学とC言語の型の違い



## 本演習で用いる変数の型 (スタイル規則参照)

整数(離散量)

int

年齢、  
日数、等

実数(連続量)

double

体重、  
温度、等

明確に区別すること。

11

## 文字と文字列

char



char型には、半角文字1文字だけを保存できる。

char



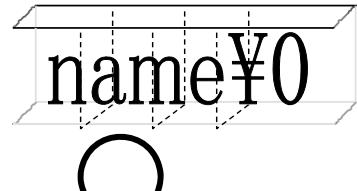
char



char



char \*



C言語では、文字列は終端文字¥0で終わる。

12

## 変数宣言(1)

[規則] Cでは使用する変数をすべて宣言しなければならない。

### 変数宣言書式

```
データ型1 変数名1;  
データ型2 変数名2;
```

### 変数宣言例1

```
int age; /* 年齢 */
```

変数宣言は、プログラム内で用いるデータを入れる入れ物(変数)を用意して、その入れ物に名前をつけると考えればよい。  
宣言の際には、必ずコメントを付けること。  
(スタイル規則参照)

13

## 変数宣言(2)(同型複数の変数宣言)



### 変数宣言例2

```
double x_pos; /* x座標 */  
double y_pos; /* y座標 */
```

14

## 変数宣言(3)(異なる型の変数宣言)

### 変数宣言例3

```
int age;           /* 年齢 */
double x_pos;     /* x座標 */
double y_pos;     /* y座標 */
```

15

## 宣言場所(1)

[規則]変数宣言は、関数の最初でまとめて行う。

### 典型的なmain関数

```
int main()
{
    /*変数宣言*/
    int age;           /* 年齢 */
    double x_pos;     /* x座標 */
    double y_pos;     /* y座標 */

    /*変数初期化*/
    .....
}
```

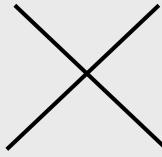
16

## 宣言場所(2)(不正な宣言)

```
int main()
{
    /*変数宣言1*/
    int age; /* 年齢 */

    /*演算1*/
    f1=0;
    .....

    /*変数宣言2*/
    double x_pos; /* x座標 */
}
```

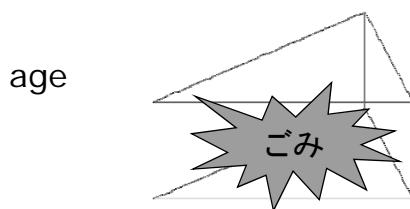


なお、main関数先頭以外での変数宣言については、  
第9回で説明。

17

## 宣言直後の変数の中身

変数は宣言しただけでは、変数内のデータは不定です。  
つまり、プログラムの実行時によって異なります。



18

## 変数への代入(1)

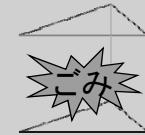
典型的なmain関数

```
int main()
{
    /*変数宣言*/
    int age; /*年齢*/

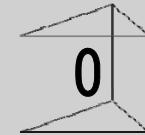
    /*変数への代入*/
    age=0;

    /* .... */
    ....
}
```

この段階での状態



この段階での状態



'='は、ソースコード内で、  
変数に値を代入する方法(演算子)。  
詳しくは次回。

19

## 変数への代入(2)

double型への代入

```
int main()
{
    /*変数宣言*/
    double weight; /*体重*/

    /*変数への代入*/
    weight=0.0;

    /* .... */
    ....
}
```

定数にも型があるので、  
注意すること。  
詳しくは、次回。  
(スタイル規則参照)



20

## 変数の中身の表示

printf文を用いて、表示できる。

” ”間の文字列に特別な文字列を挿入して、  
(カンマ)を書き、  
その後に変数名を書けばよい。

“変換仕様” =  
“%” + “変換文字”

典型的な表示書式

```
printf("……%変換文字……", 変数名);
```

21

## printf文

```
printf("You are %d years old\n", age);
```

文字列を標準出力(ディスプレイ)に出力するライブラリ関数

%変換文字 printf文の文字列内の%変換文字(変換仕様)は、  
後の変数に関する出力指示を表わす。

(int用)

%d 10進数の整数として表示

%6d 10進数として印字、少なくとも6文字幅で表示

%o 8進数の整数として表示

%x 16進数の整数として表示

(double用)

%f 小数(double)として表示

%6.2f 表示幅として6文字分とり、小数点以下2桁まで表示

%.2f 小数点以下2桁で表示

22

## printf文における変換仕様と型の対応

10進数

8進数

16進数

%d  
%o  
%x

%f

浮動小数点数

int

整数

double

実数

23

## printf文における複数の変換仕様

```
int a1;
int a2;
int a3;
a1=1;
a2=2;
a3=3;
printf("first %d second %d third %d\n",a1,a2,a3);
```

```
int a;
a=25;
printf("10進数%d 8進数 %o 16進数%x\n",a,a,a);
```

24

## printf文に関するよくある間違い

```
int age;
double weight;

age=19;
weight=63.5;

printf("My age is %d\n",age);
printf("The weight is %f\n",weight);
```

カンマ忘れ

(変換仕様と変数の  
型の不一致)

25

## 練習1

```
/* 変数の中身表示実験 print_val.c コメント省略*/
#include<stdio.h>
int main()
{
    int a;
    printf("代入前 a=%d\n",a);

    a=0;
    printf("代入後 a=%d\n",a);

    a=3;
    printf("正しい変換仕様a=%d\n",a);
    printf("変換仕様間違いa=%f\n",a);

    return 0;
}
```

## 変数に値を代入する (プログラム実行後)

演算子'='によって、  
変数に値を代入するには、  
ソースコード内に記述しないといけない。

scanf文を用いて、標準入力(キーボード)から  
変数に値を代入できる。

典型的な代入書式

```
scanf("%変換文字",&変数名);
```

27

## scanf文

```
scanf("%d ",&age);
```

標準入力(キーボード)から変数に値を読み込むライブラリ関数

%変換文字 scanf文の" "内の%で始まる文字は変換仕様である。  
scanfの" "内には変換仕様しか書かないこと。

&変数      scanf文の変数名には&をつけること。  
&は変数のアドレスを表わす。  
詳しくは、第11回ポイントで説明する。

%d 整数を入力

%lf 小数を入力(double型)

printfの変換文字との  
相違に注意が必要。

28

## scanf文における変換文字と型の対応

10進数

%d

int

整数

%lf

double

実数

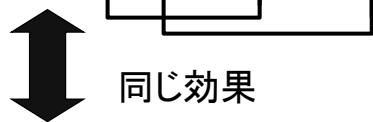
浮動小数点数  
(printf文の変換仕様との  
違いに注意)

29

## scanf文における複数の変換文字

```
char initial;  
int age;  
double weight;
```

```
scanf("%d%lf", &age, &weight);
```



```
/*同じ宣言*/  
scanf("%d", &age);  
scanf("%lf", &weight);
```

30

### scanf文に関するよくある間違い

```
int age;  
double weight;
```

&忘れ

```
scanf("%d",age);  
scanf("%lf",&weight);  
scanf("%d",&weight);  
scanf("%6.2lf",&weight);  
scanf("%1f",&weight);
```

カンマ忘れ

型の不一致

printf文用の  
変換仕様の  
誤用

'1'と'l'の間違い

31

### 標準入出力

UNIXのコマンドは、通常、標準入力と標準出力を用いる。

標準入力：通常はキーボードだが、  
リダイレクション'<'を用いて  
ファイルに変更可能。

標準出力：通常は画面だが、  
リダイレクション'>'を用いて  
ファイルに変更可能。

32

## 練習2

```
/*標準入出力実験 test_stdio.c */
#include<stdio.h>
int main()
{
    int a;
    int b;
    int c;

    scanf("%d%d%d",&a,&b,&c);
    printf("a=%d b=%d c=%d\n",a,b,c);

    return 0;
}
```

33

## 標準入力の変更 (ファイルから入力する)

実行

```
$ ./実行ファイル名 < 入力ファイル名  
(データファイル)
```

実行例

```
$./test_stdio < test_stdio.in  
1 3 5  
$
```

データファイル  
test\_stdio.in

```
1  
3  
5
```

34

## 標準出力の変更 (ファイルへ出力する) 実行

```
$ ./実行ファイル名 > 出力ファイル名  
(空ファイルでいい。)
```

### 実行例

```
./test_stdio > test_stdio.out  
2  
4  
6  
$lv test_stdio.out  
a=2 b=4 c=6
```

35

## 標準入出力の同時変更

リダイレクションを組み合わせればいい。

```
$ ./実行ファイル名 < 入力ファイル > 出力ファイル
```

```
./test_stdio <test_stdio.in > test_stdio.out  
$lv test_stdio.out  
a=1 b=3 c=5
```

36

## 入出力のあるプログラム例 (教科書p.36参照)

```
/*
 作成日:yyyy/mm/dd
 作成者:本荘 太郎
 学籍番号:B0zB0xx
 ソースファイル:echoage.c
 実行ファイル:echoage
 説明:入力された年齢を表示するプログラム
 入力:標準入力から年齢を入力する。
 出力:標準出力に年齢を出力する。
*/
/* プログラム本体は次のページ */
```

37

```
/* 前ページのプログラムの続き */
#include <stdio.h>
int main()
{
    /* 変数宣言 */
    int age; /*年齢*/

    /* 入力処理 */
    printf("年齢は?¥n");
    scanf("%d",&age);

    /* 出力処理 */
    printf("年齢は %d 歳です。¥n",age);
    return 0;
}
```

## 実行結果

```
$make  
gcc echoage -o echoage  
$ ./echoage  
年齢は?  
20  
年齢は 20 歳です。  
$
```

年齢は?  
20

年齢は 20 歳です。

キーボードから  
打ち込む