第1週
ファイル入出力と構造体

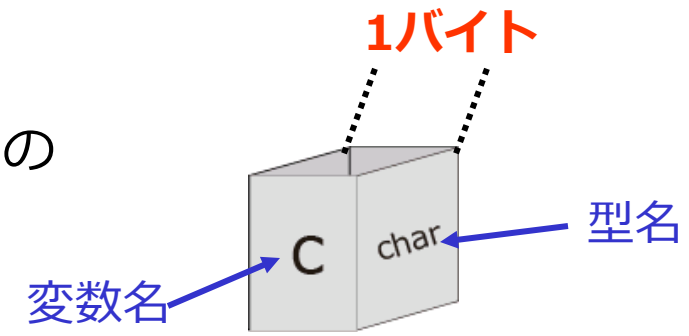
習得すること

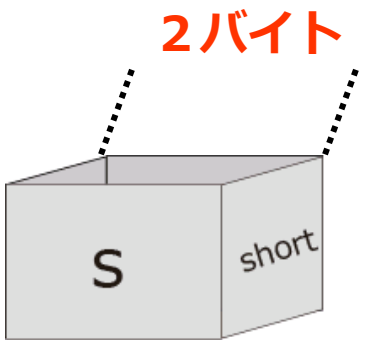
- 復習
  - 変数
  - 配列
- scanf
- ファイル
  - fopen,
    - オープンモード
  - fscanf, fprintf
  - fclose
- 構造体
  - 構造体配列
  - 構造体初期値設定
- ユーザ定義型　typedef

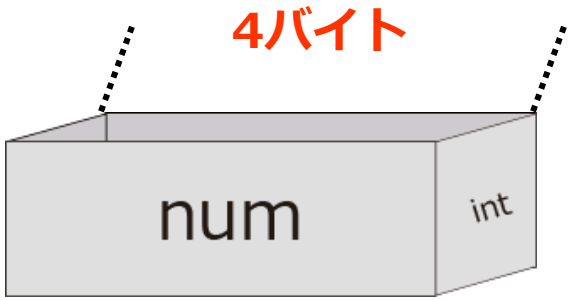
復習：変数

●変数

- 数値や文字を格納する箱のようなもの
- 型によってサイズが違う
  - char型：1 バイト
  - short型：2バイト
  - int型：4バイト（環境によって異なる）





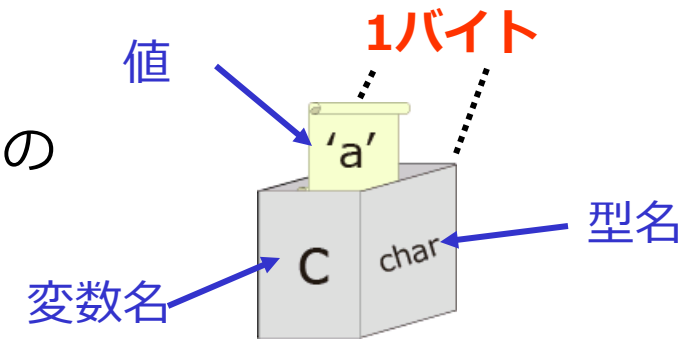


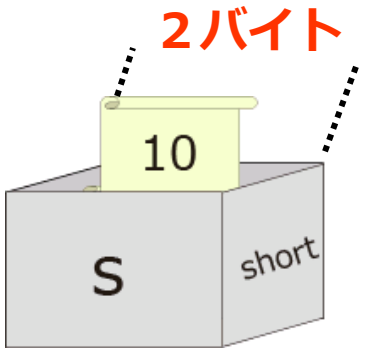

```
int main(void){
  char c;
  short s;
  int num;
  ...
  return 0;
}
```

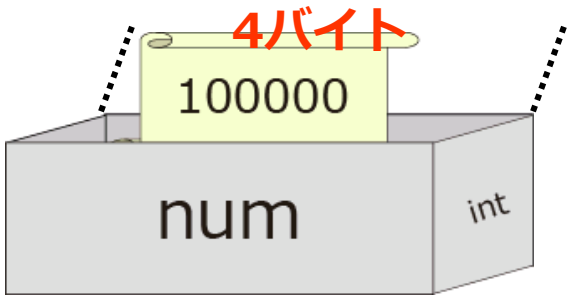
復習：変数

●変数

- 数値や文字を格納する箱のようなもの
- 型によってサイズが違う
  - char型：1 バイト
  - short型：2バイト
  - int型：4バイト（環境によって異なる）







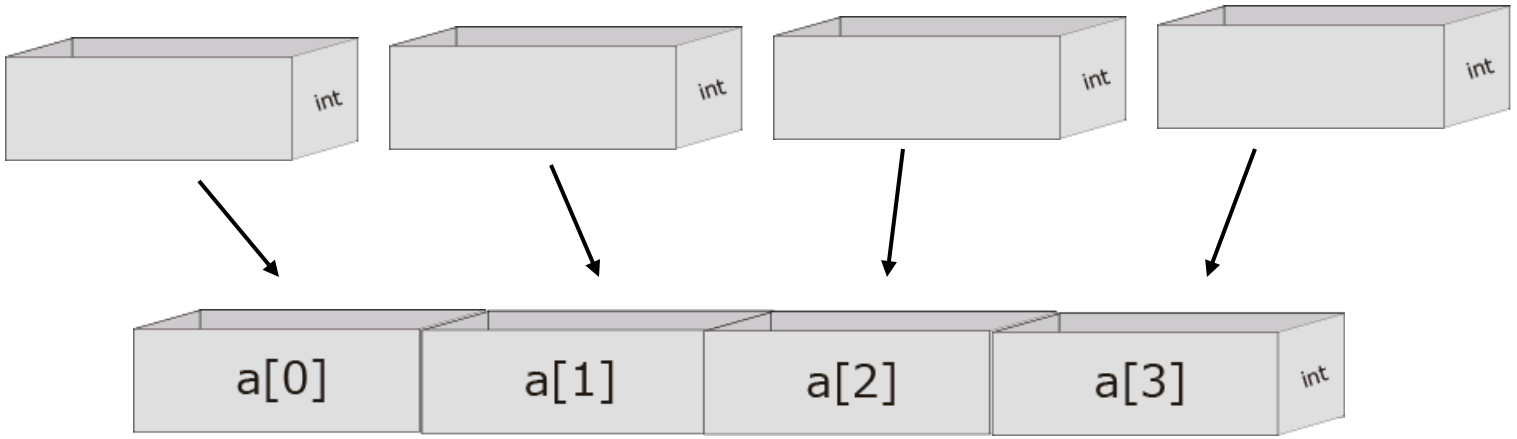

```
int main(void){
  char c ='a';
  short s = 10;
  int num = 100000;
  ...
  return 0;
}
```

復習：配列

●配列

- 複数の**同じ型**の変数をまとめたもの

```
int     a[ 4 ];
型    名前  要素数
```



scanf (複数の変数の読込み方)

intの場合

```
int main(void){
  int a, b;
  scanf(“%d”,&a);
  scanf(“%d”,&b);
}
```

同じ

```
int main(void){
  int a, b;
  scanf(“%d %d”,&a, &b);
}
```

文字列以外の変数には＆を付ける
（文字列には＆は必要ない）

文字列の場合

```
int main(void){
  char s1[ 10 ], s2[ 10 ];
  scanf(“%s”,s1);
  scanf(“%s”,s2);
}
```

同じ

```
int main(void){
  char s1[ 10 ], s2[ 10 ];
  scanf(“%s %s”,s1, s2);
}
```

２個以上でもまとめることができる

scanf (複数の変数の読込み方)

intと文字列の場合

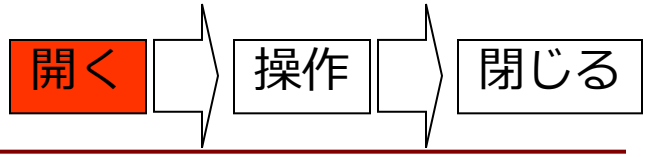
```
int main(void){
  int a;
  char s[ 10 ];
  scanf(“%d”,&a);
  scanf(“%s”,s);
}
```


```
int main(void){
  int a;
  char s[ 10 ];
  scanf(“%d %s”,&a, s);
}
```

２個以上でもまとめることができる

fscanfも同様の記述ができる

ファイル:ファイルオープン



ファイルポインタの宣言　注：ポインタは来週学びます

```
FILE *fp;
fp = fopen( “indata.data” , ”r” );
```

ファイルの開き方

fopen(**“ファイル名”**, **”オープンモード”**);

オープンモード
“r” :読み出し専用
“w” :書込み専用
　　（すでにあるファイルはなくなるので注意）
“a” :追加書込み

他にもありますが、
とりあえずこの3つを使えるようにしましょう

ファイル:エラーの確認（ファイルが開けたか？）

```
fp = fopen(“indata.data”,”r”);

if( fp == NULL ){
   //エラーメッセージ
    //エラー処理
}
```

ファイルが開けなかった場合には
fpにはNULLが代入されます

printfなどでエラーメッセージを出力

return 1;やexit(1);を使うことが多い

代入時にエラーの確認もできます

```
if( (fp = fopen(“indata.data”,”r”)) == NULL){
    //エラーメッセージ
    //エラー処理
}
```

ファイル：読込み

開く ⇨ 操作 ⇨ 閉じる

```
FILE *fp;
int num;
if( (fp = fopen(“indata.data”, ”r”)) == NULL){
   //エラーメッセージ
    //エラー処理
}

fscanf(fp, “%d”, &num );
…
```

オープンモードを
読込み専用（“ｒ”）にする

ファイルの読込み

fscanf(**ファイルポインタ**，scanfと同じ形式);

ファイル入力の終了判定

開く ⇨ 操作 ⇨ 閉じる

ファイルの入力データの終わりの場合は、
fscanf関数では戻り値EOFを返す

```
while( fscanf(fp, “%d”, &num ) != EOF){
  …
}
```

EOF(End Of File)

他の終了判定方法は
feof関数がある
書式　　int feof(FILE* fp)
戻り値 0：ファイル終端ではない
　　0以外：ファイル終端

ファイル：ファイルへ出力

開く ⇨ 操作 ⇨ 閉じる

```
FILE *fpout;
if( (fpout = fopen(“outdata.data”, ”w”)) == NULL){
    //エラーメッセージ
    //エラー処理
}

fprintf(fpout,“Output test.¥n”);
…
```

オープンモードを
書込み専用（“w”）にする

ファイルへの出力

fprintf(**ファイルポインタ**，printfと同じ形式);

ファイル:ファイルのクローズ

開く ⇨ 操作 ⇨ **閉じる**

```
FILE *fp;
int num;
if( (fp = fopen(“indata.data”,”r”)) == NULL){
   //エラーメッセージ
    //エラー処理
}

//ファイルを利用した処理
fclose(fp);
```

← ファイル利用後には必ずファイルを閉じる

ファイルを閉じる

fclose(**ファイルポインタ**);

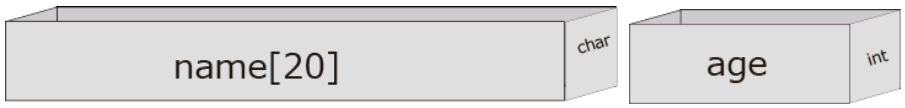
復習：構造体：構造体とは？

●構造体

- 複数の**型**の変数をまとめたもの

```
char name[20];
int  age;
```

```
struct Person{
  char name[20];
  int  age;
};
```



復習：構造体：構造体の定義

●構造体

- 複数の**型**の変数をまとめたもの

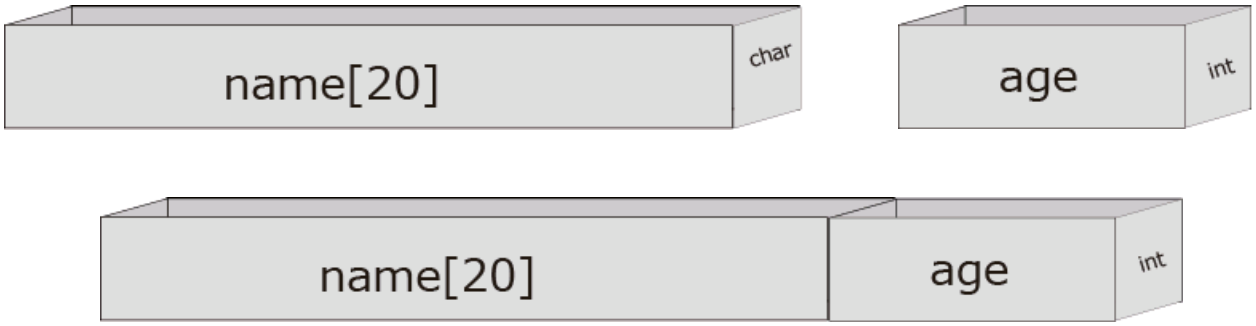
```
struct Person{
  char name[20];
  int  age;
};
```

構造体内の変数を
構造体のメンバと呼ぶ

構造体の定義

```
struct 構造体名{
  型名   メンバ名;
  型名   メンバ名;
};
```

必要な回数
型名　メンバ名；
を繰り返す



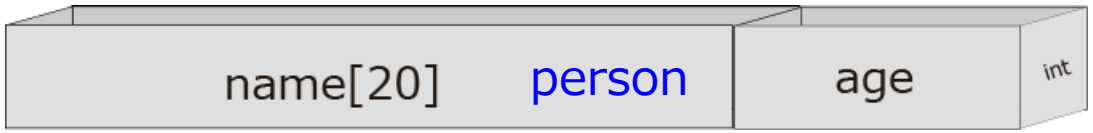
復習：構造体：構造体の宣言

構造体の宣言

struct 構造体名 変数名;

```
struct Person person;

person.age = 30;
strcpy(person.name,”Takuya Azumi”);
```

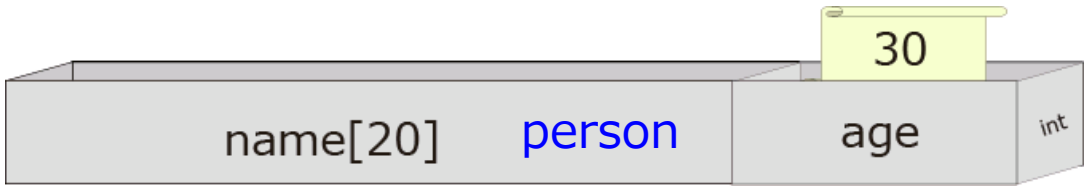


復習：構造体：構造体メンバへのアクセス

```
struct Person person;

person.age = 30;
strcpy(person.name,”Takuya Azumi”);
```

メンバのアクセス方法
構造体の変数名.メンバ名



復習：構造体：構造体メンバへのアクセス

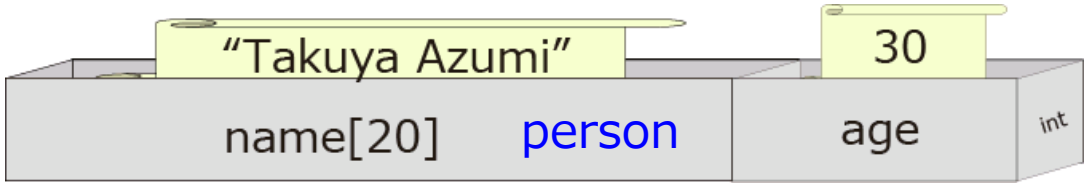
メンバのアクセス方法

構造体の変数名.メンバ名

```
struct Person person;

person.age = 30;
strcpy(person.name,”Takuya Azumi”);
```

strcpyは文字列のコピーを行う関数



構造体：構造体の初期化（宣言時）

nameに格納　　ageに格納

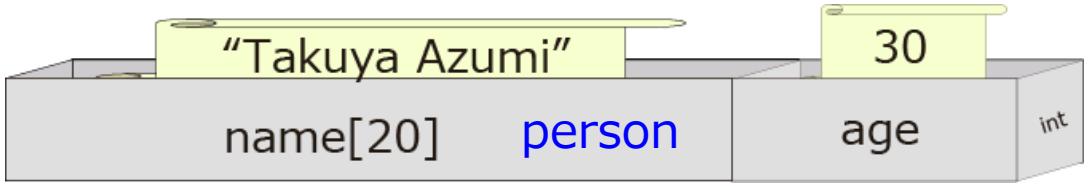
```
struct Person person = {“Takuya Azumi”, 30};
```

```
struct Person{
  char name[20];
 int  age;
};
```

構造体の初期化

struct 構造体名 変数名 = {値,　値} ;

値の順番はメンバ宣言の順番と同じ



構造体：構造体配列

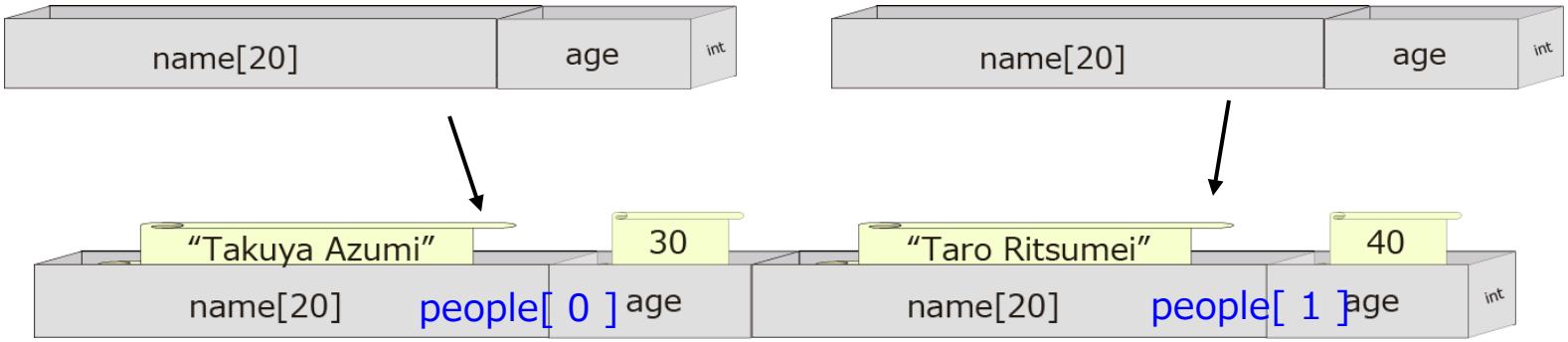
●構造体も配列にできます

```
struct Person people[ 2 ];
```

構造体配列の宣言

struct 構造体名 変数名[要素数];

```
//構造体配列のアクセスの方法
people[ 0 ].age = 30;
strcpy(people[ 0 ].name,"Takuya Azumi");
people[ 1 ].age = 40;
strcpy(people[ 1 ].name,"Taro Ritsumei");
```



構造体：typedef:新しい型名を付ける

```
typedef unsigned int uint;
uint data;
```

typedefの使い方

typedef　型の定義　新しい型名;

typedefを利用する場合は、
ここの名前を省略可

```
typedef struct Person {
  char name[20];
 int  age;
}PERSON;
```

新しい型名

```
PERSON person;
```

どちらも同じ意味になります

```
struct Person person;
```

著者リスト

1. 安積 卓也（情報システム学科）