応用プログラミング

第3回　型変換、入出力処理

メディア情報コース
平居　悠（ひらい　ゆたか）

到達目標

- C言語を通じてコンピュータの根幹部分を理解する。
- 目的に応じたプログラムを自由に設計できる。

前回

第１回　C言語の基本文法・言語処理系
第２回　数学的計算、配列、文字列
第３回　型変換、入出力処理
第４回　デバッグ、関数定義、変数の分離
第５回　プロトタイプ宣言、分割コンパイル
第６回　配列の受け渡し、ポインタ
第７回　まとめ、期末試験

前回の目標

C言語で数学的な計算、複数の変数を持つデータの扱い、文字列の入出力ができるようになる。

前回学んだこと

1. 数学的計算
2. 配列
3. 文字列

式と演算子

演算子：特定の操作機能を有した記号 (+, −, *, /など)

オペランド：演算子の作用対象

インクリメント・デクリメント演算子

インクリメント演算子 (++): 値を1増やす
デクリメント演算子 (--):値を1減らす

不等号演算子

> 左辺は右辺より大きい
< 左辺は右辺より小さい
>= 左辺は右辺より大きいまたは等しい
<= 左辺は右辺より小さいまたは等しい

等価演算子と非等価演算子

等価演算子 (==):
両辺が等しいときに真 (1)
非等価演算子 (!=):
両辺が等しくないときに真 (1)

代入演算子

単純代入演算子「=」

複合代入演算子:

「+=」：加算代入演算子

「−=」：減算代入演算子

「*=」：乗算代入演算子

「/=」：除算代入演算子

「%=」：剰余代入演算子

配列の基本

配列：変数の集まり

宣言方法：

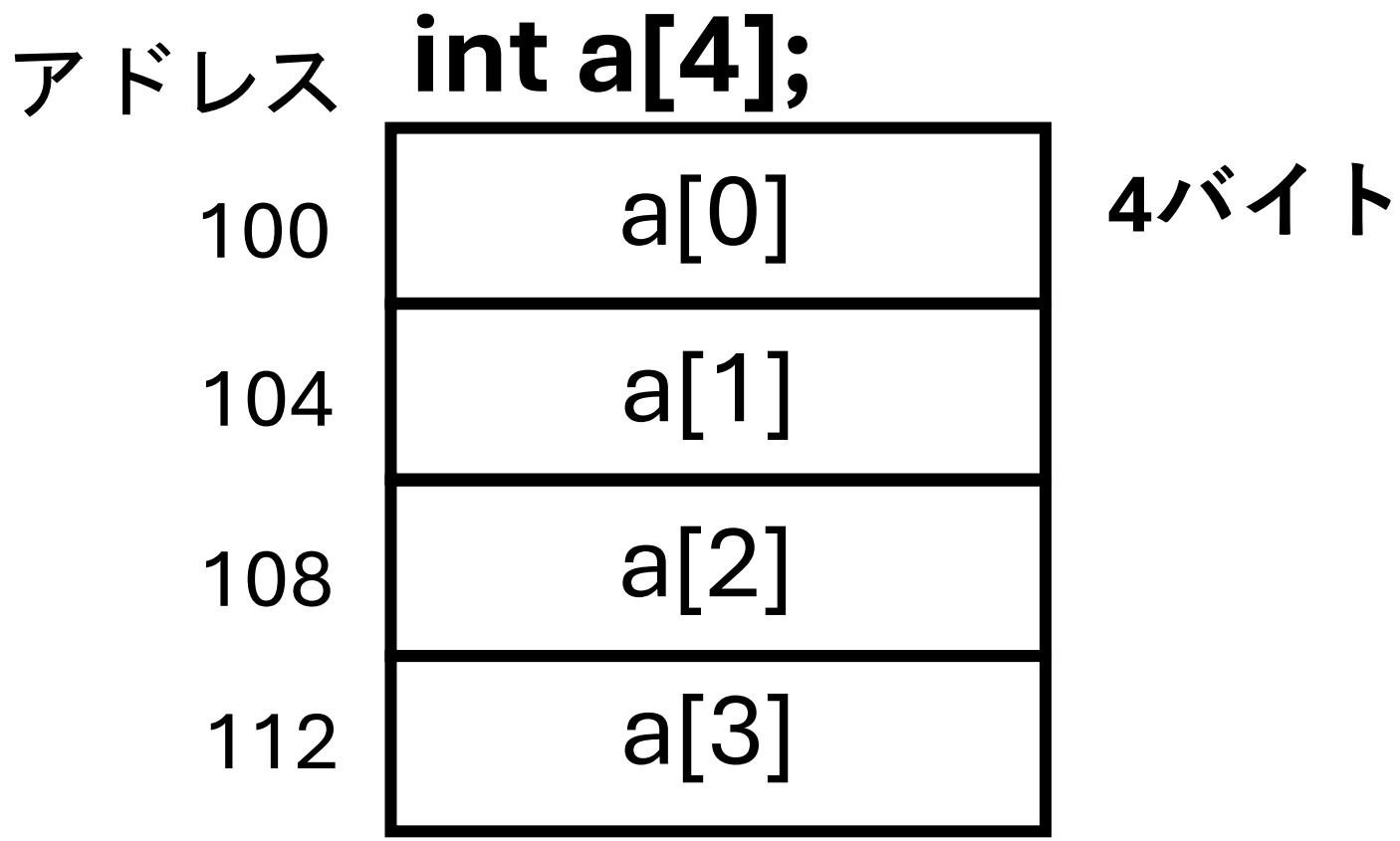
データ型　配列名[配列の大きさ];

例：

int a[4];

配列とアドレス

a[0], a[1], a[2], a[3]の順でメモリ上に隙間なく並んでいる



配列のメモリ内の状態

多次元配列

次元が２以上の配列

例：

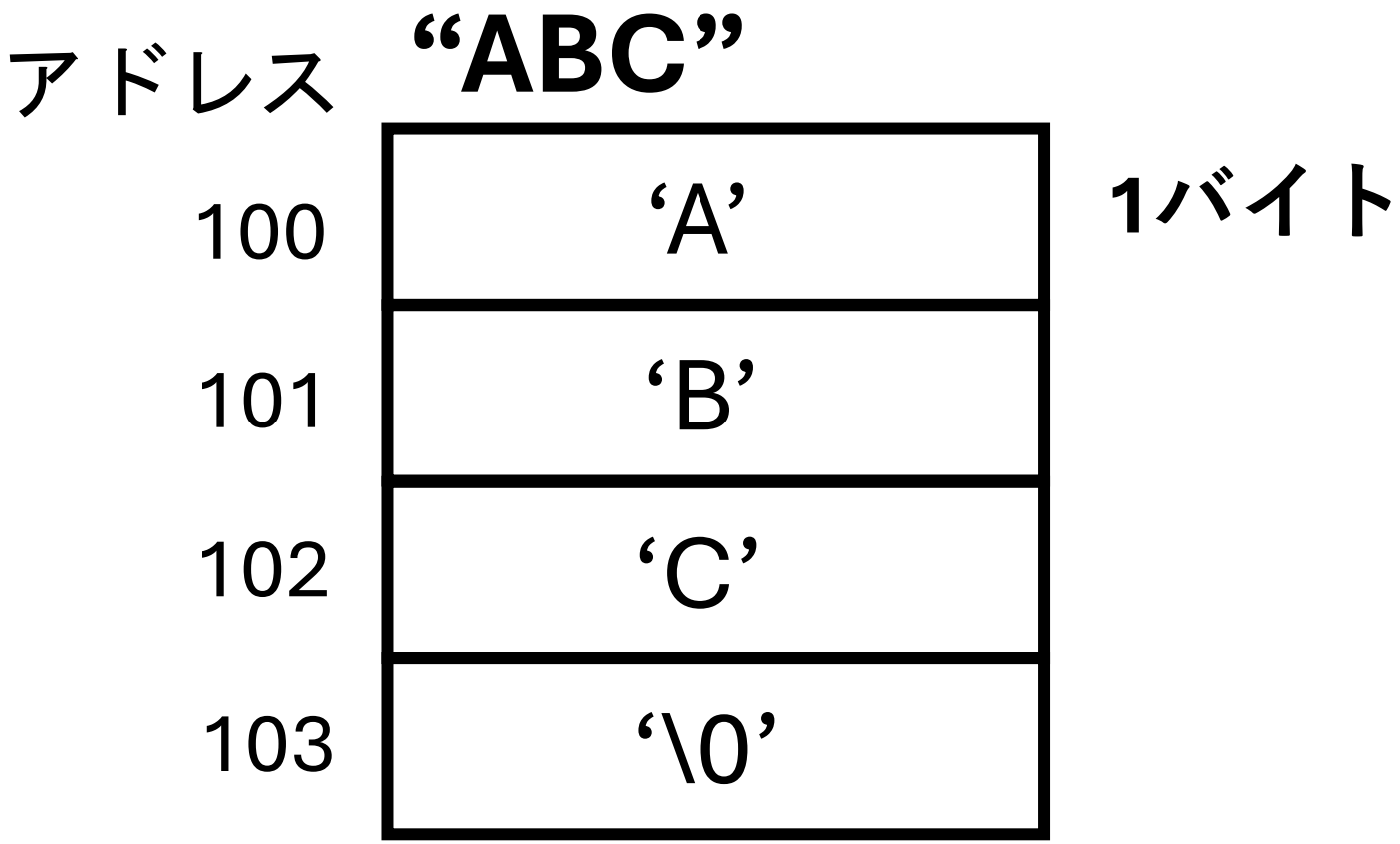
```
int a[5][2];
int a[5][2][6];
```

文字列

- ABCのような**文字列**という。
- “ABC”とダブルクォーテーションで囲ったものを**文字列リテラル**という。
- printf関数で文字列出力するときの変換指定子は**%s**

文字列とアドレス

- 文字列の式の値は先頭文字の置かれたアドレス
- 文字配列最後の要素は必ず **‘\0’ (ヌル文字**)



配列のメモリ内の状態

課題

以下の表はTaroさん、Hanakoさん、Takashiさんの英語と数学のテストの点数である。配列を用いて３名の名前と英語・数学の点数、平均点をそれぞれ表示するプログラムを作成せよ。

名前	英語	数学
Taro	90	60
Hanako	70	80
Takashi	50	40

提出先：yutaka.hirai@koeki-u.ac.jp

件名：応用プログラミング第2回課題 [学籍番号]

締め切り：10月15日 (木)

前回の問い

- C言語での演算子にはどのような規則になっているか？
  - 演算子はオペランドを伴って式を構成し、優先順位が決められている。
- 配列とは何か？
  - 配列は変数の集まりである。
- C言語で文字列はどのようにして扱うのか？
  - 文字列も配列として扱うことができる。

今回

第１回　C言語の基本文法・言語処理系
第２回　数学的計算、配列、文字列
第３回　型変換、入出力処理
第４回　デバッグ、関数定義、変数の分離
第５回　プロトタイプ宣言、分割コンパイル
第６回　配列の受け渡し、ポインタ
第７回　まとめ、期末試験

今回の目標

C言語でデータ型変換、ファイル入出力処理ができるようになる。

今回の問い

- 宣言したデータ型と異なるデータ型で計算したい場合はどうすれば良いか？
- C言語でファイル入出力はどう処理するのか？

今回学ぶこと

1. 型変換
   1. データ型
   2. キャスト
2. 入出力処理
   1. ファイル入出力
   2. バイナリファイル

今回学ぶこと

1. 型変換
    1. データ型
    2. キャスト
2. 入出力処理
    1. ファイル入出力
    2. バイナリファイル

３つのデータ型

- 整数型
- 浮動小数点数型
- 文字型

整数型

- short型 (16ビット)
- int型 (32ビット)
- long型 (32ビット)

サイズ（コンパイラにより異なる）：

short ≤ int ≤ long

int型のサイズ

現在のコンパイラでは**32ビット**の場合が多い。32ビットの場合格納できる整数の範囲は、

-2^{31}~$2^{31}-1$

−21億4748万3648 ~ 21億4748万3647

浮動小数点数型

- float型 (32ビット)
- double型 (64ビット)
- long double型 (64ビット)

サイズ（コンパイラにより異なる）：

float ≤ double ≤ long double

文字型

char型

(コンパイラによらず**1**バイト)

データ型まとめ

データ型には主に整数型 (short, int, long)、浮動小数点数型 (float)、文字型 (char) がある。

今回学ぶこと

1. 型変換
   1. データ型
   2. キャスト
2. 入出力処理
   1. ファイル入出力
   2. バイナリファイル

average01.cというファイルを作成し、実行してみよう

```
#include <stdio.h>

int main()
{
   int score[3] = {90, 60, 50};
   int i;
   int sum = 0;

   for(i = 0; i < 3; i++){
      sum += score[i];
   }

   printf("平均点は%d点です。\n", sum/3);
   return 0;
}
```

出力結果

平均点は66点です。

でも

(90+60+50)/3 = 66.66666....

と割り切れないはず。

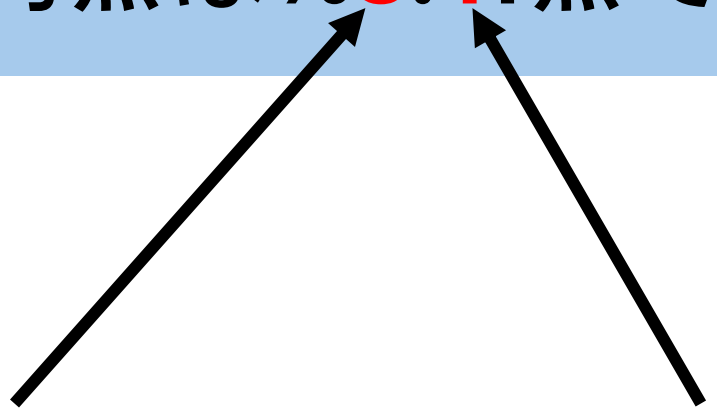
int型同士の割り算

- int型同士の割り算はint型 (整数)になる。
- 小数点以下は切り捨てられる。

average01.cを右のように書き換えてみよう。

```
#include <stdio.h>

int main()
{
   int score[3] = {90, 60, 50};
   int i;
   int sum = 0;

   for(i = 0; i < 3; i++){
      sum += score[i];
   }

   printf("平均点は%3.1f点です。\n",
sum/3.0);
   return 0;
}
```

```
printf("平均点は%3.1f点です。\n", sum/3.0);
```



小数点含めて全体の桁は３桁

小数点以下は１桁

出力結果

平均点は66.7点です。

異なるデータ型同士の計算では、値は大きいデータ型に合わせられる。

キャスト演算子

式の型を変更する。

(型名X)式

average01.cを右のように書き換えてみよう。

```
#include <stdio.h>

int main()
{
   int score[3] = {90, 60, 50};
   int i;
   int sum = 0;

   for(i = 0; i < 3; i++){
      sum += score[i];
   }

   printf("平均点は%3.1f点です。\n", (double)sum/3);
   return 0;
}
```

今回の問い

- 宣言したデータ型と異なるデータ型で計算したい場合はどうすれば良いか？
  - キャスト演算子でデータ型を変更する。
- C言語でファイル入出力はどう処理するのか？

今回学ぶこと

1. 型変換
   1. データ型
   2. キャスト
2. 入出力処理
   1. ファイル入出力
   2. バイナリファイル

ファイル入出力

- ファイルは**fopen**関数で開き、**fclose**関数で閉じる。
- ファイルへの出力は**fprintf**関数を用いる。

ファイル出力

file01.cというファイルを右のように作成し、実行しよう。

```
#include <stdio.h>

int main()
{
   FILE *fp;
   fp = fopen(“test.txt”, “w”);

   if (fp == NULL) {
      perror(“ファイルを開けませんでした。\n”);
      return -1;
   } else {
      printf(“ファイルを正常に開きました。\n”);
   }

   fprintf(fp, “初めてのファイル入出力です。\n”);

   if (fclose(fp) != 0) {
      perror(“ファイルを閉じられませんでした。\n”);
      return -1;
   } else {
      printf(“ファイルを正常に閉じました。\n”);
   }

   return 0;
}
```

ファイルポインタ

FILE *fp;

ファイルを読み込むメモリの
アドレス値を指定

fopen関数

```
fp = fopen(“test.txt”, “w”)
```

ファイルを書き込みモード(**“w”**)で開く。読み込みモードは(**“r”**)。

ファイルオープンエラー処理

```
if (fp == NULL){
   perror(“ファイルを開けませんでした。\n”)
   return -1;
}
```

ファイルが存在しない場合、エラーメッセージ「ファイルが開けませんでした。::No such file or directly」と出力してプログラムを終了する。

fprintf関数

fprintf(fp, “初めてのファイル入出力です。\n”)

最初の引数はファイルポインタ (fp)。他はprintf関数と同じ。この場合、test.txtに「初めてのファイル入出力です。」と出力される。

cat text.txtとターミナルに入力して、test.txtの中身を確認してみよう。

fclose関数

```
if (fclose(fp) != 0) {
   perror(“ファイルを閉じられませんでした。\n”);
   return -1;
}
```

ファイルを閉じる関数。成功した時は0が返され、失敗した時はEOFが返される。

ファイル入力

file02.cというファイルを右のように作成し、実行しよう。

```
#include <stdio.h>

int main()
{
   FILE *fp;
   char text[256];

   fp = fopen(“test.txt”, “r”);

   if (fp == NULL) {
      perror(“ファイルを開けませんでした。\n”);
      return -1;
   }

   while (fscanf(fp, “%s”, text) != EOF)
      printf(“%s\n”, text);

   if (fclose(fp) != 0) {
      perror(“ファイルを閉じられませんでした。\n”);
      return -1;
   }

   return 0;
}
```

fscanf関数

fscanf(fp, “%s”, text)

最初の引数はファイルポインタを指定。ファイルの終端まで来たらEOFを返す。

ファイル入力

file03.cというファイルを右のように作成し、実行しよう。

```
#include <stdio.h>

int main()
{
   FILE *fp;
   int c;

   fp = fopen(“test.txt”, “r”);

   if (fp == NULL) {
      perror(“ファイルを開けませんでした。\n”);
      return -1;
   }

   while ((c = fgetc(fp)) != EOF)
      putc(c, stdout);

   if (fclose(fp) != 0) {
      perror(“ファイルを閉じられませんでした。\n”);
      return -1;
   }

   return 0;
}
```

fgetc関数

fgetc(fp)

ファイルから1文字ずつ読み込み、int型で返す。ファイルの終端まで来たらEOFを返す。

putc関数

putc(c, stdout)

ファイルに文字を出力する。標準出力（画面）に出力する場合は**stdout**を指定する。

今回学ぶこと

1. 型変換
    1. データ型
    2. キャスト
2. 入出力処理
    1. ファイル入出力
    2. バイナリファイル

バイナリファイルとは

コンピュータ内部で使用されるデータ形式をそのまま書き込んであるファイル。データサイズを小さくでき、処理速度も速い。

バイナリファイル入出力

1. fopen関数でファイルを開く。読み込みモードは“**rb**”、書き込みモードは“**wb**”。
2. 書き込みには**fwrite**関数、読み込みには**fread**関数を使う。
3. 読み書きが終了したらfclose関数でファイルを閉じる。

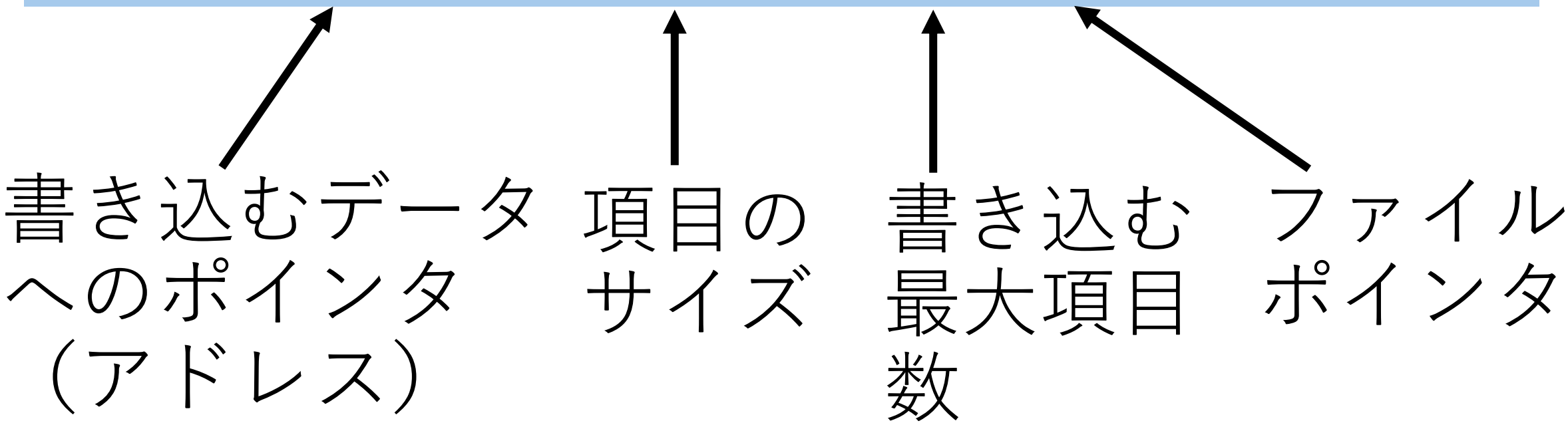
ファイル出力

binary01.cというファイルを右のように作成し、実行しよう。

```
#include <stdio.h>

int main()
{
   FILE *fp;
   int a[3] = {1, 2, 3};

   fp = fopen(“mybinary.b”, “wb”);
   if (fp == NULL) {
      perror(“ファイルを開けませんでした。\n”);
      return -1;
   }

   fwrite(a, sizeof(int), 3, fp);

   if (fclose(fp) != 0) {
      perror(“ファイルを閉じられませんでした。\n”);
      return -1;
   }

   return 0;
}
```

fwrite関数

```
fwrite(a, sizeof(int), 3, fp);
```



Emacsバイナリファイル編集モード

Emacsでバイナリファイルを開くには、バイナリ編集モード(**Hexlモード**)を用いる。
[**Alt**] + [**x**]または[**Esc**] + [**x**]キーを押してから、「**hexl-find-file**」と入力して[**Enter**]キーを押すと、

Filename: ~/

と表示されるので、ファイル名を入力して[**Enter**]キーを押す。

バイナリファイルの読み方

0100 0000 0200 0000 0300 0000

int型は4バイトなので、4バイトを１かたまりとして見ると、

01 00 00 00

02 00 00 00

03 00 00 00

となる。インテル製のCPUではバイトオーダーはリトルエンディアン（最も小さい桁から順にメモリに格納）となっている。

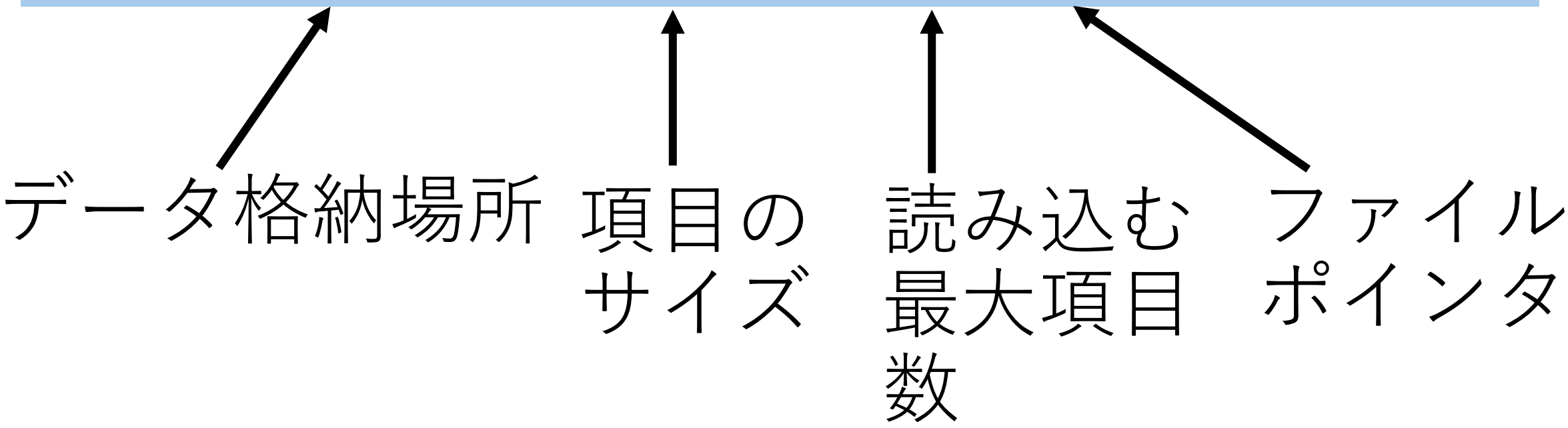
ファイル入力

binary02.cというファイルを右のように作成し、実行しよう。

```
#include <stdio.h>

int main()
{
   FILE *fp;
   int a[3] = {1, 2, 3};
   int i;

   fp = fopen(“mybinary.b”, “rb”);
   if (fp == NULL) {
      perror(“ファイルを開けませんでした。\n”);
      return -1;
   }

   fread(a, sizeof(int), 3, fp);

   if (fclose(fp) != 0) {
      perror(“ファイルを閉じられませんでした。\n”);
      return -1;
   }

   for(i = 0; i < 3; i++)
      printf(“%d\n”, a[i]);

   return 0;
}
```

fread関数

```
fread(a, sizeof(int), 3, fp);
```



今回の問い

- 宣言したデータ型と異なるデータ型で計算したい場合はどうすれば良いか？
  - キャスト演算子でデータ型を変更する。
- **C言語でファイル入出力はどう処理するのか？**
  - **ファイルを開くにはfopen関数、ファイルを閉じるにはfclose関数を用いる。**

課題

第2回課題で画面に出力したテストの点数と各教科の平均点を記述したファイル、results.txtを出力するプログラム、results.cを作成せよ。ただし、Hanakoの英語と数学の点数はそれぞれ75点、90点に変更する。平均点は小数第一位まで出力せよ。生成AIでのコード生成は不可。

提出先：yutaka.hirai@koeki-u.ac.jp

件名：応用プログラミング第3回課題 [学籍番号]

提出物：results.c

締め切り：10月22日 (木)

今回学んだこと

1. 型変換
   1. データ型
   2. キャスト
2. 入出力処理
   1. ファイル入出力
   2. バイナリファイル

今回の目標

C言語でデータ型変換、ファイル入出力処理ができるようになる。

次回

第１回　C言語の基本文法・言語処理系
第２回　数学的計算、配列、文字列
第３回　型変換、入出力処理
第４回　デバッグ、関数定義、変数の分離
第５回　プロトタイプ宣言、分割コンパイル
第６回　配列の受け渡し、ポインタ
第７回　まとめ、期末試験