2008 年度秋期

情報処理IV

(数学科)

～ データ構造とアルゴリズム～

(担当: 角皆・渋谷)

- **データ構造**
  扱う対象を、如何に捉え、
  如何に計算機上に実現するか

- **アルゴリズム (算法)**
  行なう操作を、如何に捉え、
  如何に計算機上で実行するか

⟶ 実現効率も問題となる
· · · 計算の手間 (時間)・必要メモリ量 (空間)

- **データ構造**
  扱う対象を、如何に捉え、
  如何に計算機上に実現するか

- **アルゴリズム (算法)**
  行なう操作を、如何に捉え、
  如何に計算機上で実行するか

⟶ 実現効率も問題となる
· · · 計算の手間 (時間)・必要メモリ量 (空間)

授業内容の予定:

- **変数領域の動的確保**
  (C 言語の文法事項の補足)
- **種々のデータ構造**
  (リスト・スタック・キュー・木など)
- **再帰呼出と帰納法**
- **種々のアルゴリズム (符号とその実装)**
- **計算量について**
- **分割コンパイル・make による**
  プロジェクト管理 (時間の余裕があれば)
- **総合演習 (多項式簡易電卓の作成など)**

実習 (復習)1.0:

整数成分の 3 次元ベクトルを 2 つ考え、

その成分を入力し、

その和を計算して表示する

プログラムを作成せよ。

出来れば

- ベクトルの成分を入力する
- ベクトルの和を計算する
- ベクトルを表示する

をそれぞれ個別の関数として作れ。

今は始めから
ベクトルの成分が 3 個と決まっていた。

ベクトルの成分の個数も実行中に入力したい。

(成分が何個でも対応できるプログラムを
作りたい)

(C 言語のプログラムでは)

必要な変数や配列は全てその冒頭で宣言し、

特に配列に関しては、

その大きさも予め決めておく必要があった。

(変数の宣言 $\cdots$ その変数のための領域確保)

$\longrightarrow$ 可変長に対応するには ?

可変長に対応する一法:

予め余裕を見て、

想定する充分な個数の要素を持つ

大きな配列を用意する

$\longrightarrow$ **実習 1.1**

途中まで作ったものを用意してあるので、
残り (関数の部分) を完成せよ。

予め余裕を見て
想定する充分な個数の要素を持つ
大きな配列を用意

$\longrightarrow$ 無駄が多い (かもしれない)

必要な大きさが決まってから
必要なだけ領域確保したい

$\longrightarrow$ メモリの動的確保
(dynamic memory allocation)

実習 2.1

最初にデータの個数を入力して、

その分の配列のメモリ領域を**動的に確保**し、

続いてその個数分のデータ (整数) を入力すると

その値を配列に格納した後、

その総和を表示するプログラム

メモリの動的確保: 関数 malloc() を用いる

(1) その型へのポインタ変数を予め宣言

```
type *p;
```

(2) 関数 malloc() でメモリ領域を確保し、
その先頭アドレスをポインタ変数に代入

```
p=(type *)malloc(sizeof(type)*n);
```

(3) そのポインタ変数を用いて、
確保した領域を間接参照

```
p=(type *)malloc(sizeof(type)*n);
```

(a) その変数のサイズ (メモリ量) を求める

(b) 必要な個数を掛けて、
確保するメモリ領域量を決定

(c) 所定の大きさのメモリ領域を確保して、
その先頭アドレスを返値として返す

(d) 適切な型へのポインタにキャスト
(明示的な型変換)

(e) 確保した領域の先頭アドレスを覚えておく

```
p=(type *)malloc(sizeof(type)*n);
```

(a) その変数のサイズ (メモリ量) を求める

(b) 必要な個数を掛けて、
確保するメモリ領域量を決定

(c) 所定の大きさのメモリ領域を確保して、
その先頭アドレスを返値として返す

(d) 適切な型へのポインタにキャスト
(明示的な型変換)

(e) 確保した領域の先頭アドレスを覚えておく

```
p=(type *)malloc(sizeof(type)*n);
```

(a) その変数のサイズ (メモリ量) を求める

(b) 必要な個数を掛けて、
確保するメモリ領域量を決定

(c) 所定の大きさのメモリ領域を確保して、
その先頭アドレスを返値として返す

(d) 適切な型へのポインタにキャスト
(明示的な型変換)

(e) 確保した領域の先頭アドレスを覚えておく

```
p=(type *)malloc(sizeof(type)*n);
```

(a) その変数のサイズ (メモリ量) を求める

(b) 必要な個数を掛けて、
確保するメモリ領域量を決定

(c) 所定の大きさのメモリ領域を確保して、
その先頭アドレスを返値として返す

(d) 適切な型へのポインタにキャスト
(明示的な型変換)

(e) 確保した領域の先頭アドレスを覚えておく

```
p=(type *)malloc(sizeof(type)*n);
```

(a) その変数のサイズ (メモリ量) を求める

(b) 必要な個数を掛けて、
確保するメモリ領域量を決定

(c) 所定の大きさのメモリ領域を確保して、
その先頭アドレスを返値として返す

(d) 適切な型へのポインタにキャスト
(明示的な型変換)

(e) 確保した領域の先頭アドレスを覚えておく

課題 1:

`vector0.c` **を修正して、**

ベクトルの成分の個数を入力してから、

その分だけのメモリ領域を確保することにより、

無駄なくメモリを利用するようにした

プログラム `vector1.c` **を作成せよ。**

(extra feature 歓迎!!)