Zoom URL

- **第5回講義は下記URLで実施します。**
- 5/20（木） 13:30 **までにアクセスしてください。**
  （10分程前からアクセス可能になります）

https://us02web.zoom.us/j/82736949695?pwd=Qm5aOFpBbitLN3R0VmduZW1OQlA5UT09

ミーティングID: 827 3694 9695

パスワード: 550262

コンピュータサイエンスとプログラミング
第5回演習　2021年05月20日

構造体(1) (structure)

オブジェクトを思うままに定義できるようになる！

構造体って？

- 複数の属性をもつオブジェクトを定義
- オブジェクト：世の中にあるもの全て、概念
- どう定義するかはあなた次第

学生
・学籍番号
・氏名
・メールアドレス

教授
・教員ID
・氏名
・メールアドレス
・居室
・担当授業

構造体型の宣言

- 構造体がどのような型のデータから構成されるかを宣言し，構造体タグの名前を持つ構造体型が利用できるようにする
  - 構造体タグ：この宣言を参照するための名前
  - メンバ：構造体を構成する要素

```
struct 構造体タグ{
    型名   メンバ名1;
    型名   メンバ名2;
    ……
};
```

```
struct point{
    int x;    /* x座標 */
    int y;    /* y座標 */
};
```

構造体の定義と初期化

- **構造体の定義**

```
struct 構造体タグ 構造体変数名;
```

```
struct point p1, p2;
```

- **構造体の初期化**

```
struct 構造体タグ 構造体変数名
        = {メンバ1の初期値, メンバ2の初期値, ...};
```

```
struct point p1 = {1, -1};
```

初期値の個数がメンバの個数よりも少なかった場合は，残りのメンバは整数型では0，浮動小数点型では0.0で初期化される．

```
struct point p1 = {1};
```

←(1, 0)で初期化される

構造体メンバの参照と代入

- **メンバの参照**

  構造体変数名. メンバ名

  「. 」はメンバ参照演算子

  （例）構造体型pointの変数p1から原点までの距離

```
double d = sqrt(p1.x * p1.x + p1.y * p1.y);
```

- **メンバに値を代入**

  構造体変数名. メンバ名 = 式;

  （例）

```
p1.x = p1.x + 10;
p1.y = p1.y + 10;
```

- **メンバのアドレスの取得**

  &構造体変数名. メンバ名

  （例）

```
scanf("%d %d", &p1.x, &p1.y);
```

【例題 5-1】 構造体のメンバを参照するプログラム

```
#include <stdio.h>
#include <math.h>

struct point{
   int x;   /* x座標 */
   int y;   /* y座標 */
};

int main(void){
   struct point p1;
   double d;

   printf("> (x, y) ");
   scanf("%d %d", &p1.x, &p1.y);

   d = sqrt(p1.x * p1.x + p1.y * p1.y);

   printf("Distance between (%d, %d) and (0, 0) is
                                        %.2f. ¥n", p1.x, p1.y, d);
   return 0;
}
```

- 点の座標を入力し，原点からの距離を計算するプログラム
- sqrt関数を用いるためにmath.hをインクルード
- コンパイル時に-lmのオプション*注

構造体の比較

- **構造体を比較するためには，それぞれのメンバ同士を比較する必要がある．**

（例） 2**点が同一の点かどうかを調べる**

```
if (p1.x == p2.x && p1.y == p2.y){
      2点が同一の点である時の処理
}
```

（間違いの例）

```
if (p1 == p2){
      2点が同一の点である時の処理
}
```

構造体のコピー

```
struct point p1, p2={3, 2};

p1 = p2;
```

それぞれのメンバの値を代入するのと同じ

```
p1.x = p2.x;
p1.y = p2.y;
```

ただしポインタをメンバに含む場合は、
参照渡しとなるため注意！

【例題 5-2】 構造体のメンバを比較するプログラム

```
#include <stdio.h>

struct point{
   int x;  /* x座標 */
   int y;  /* y座標 */
};

int main(void){
   struct point p1 = {3, 2};
   struct point p2;

   while(scanf("%d  %d", &p2.x, &p2.y) != EOF){
      if (p1.x == p2.x && p1.y == p2.y){
         printf("(%d, %d) is correct.¥n", p2.x, p2.y);
         return 0;
      } else{
         printf("(%d, %d) is wrong.¥n", p2.x, p2.y);
      }
   }
   return 0;
}
```

- 入力された点の座標が予め決められた点と同一かどうかを判定するプログラム

ファイルやキーボードからの
複数データの入力

```
struct point;

while(scanf("%d  %d", &p1.x, &p1.y) != EOF){
      構造体p1の処理
}
```

data.txt の例

```
1 1
2 1
2 2
3 1
3 2
```

- 入力リダイレクト機能

```
% ./a.exe < data.txt
```

  - ファイルdata.txtの先頭から1行ずつデータを入力し，scanf関数の実引数に格納される．
  - ファイルの最後に到達すると，scanf関数はEOF (End of File)を返すので，while文の繰り返しが終了する．
- キーボードからのデータ入力
  - データの入力を終了する時にCtrl-Dを入力すると，scanf関数はEOFを返すので，while文の繰り返しが終了する．

配列型のメンバを持つ構造体（例）

- 宣言

```
struct callinfo{
   int history[31];  /* 日ごとの通話時間 */
   int total;        /* 通話時間の合計 */
};
```

- 初期化

```
struct callinfo c1 = {{20, 31, 12}, 0};
```

- 参照

```
x = c1.history[i];
```

- 代入

```
c1.history[i] = 12;
```

- アドレスの取得

```
&c1.history[i]
```

メンバ参照演算子はアドレス演算子よりも優先順位が高いので，`&(c1.history[i])`のように括弧で囲む必要はない．

文字列のメンバを持つ構造体（例）

- 宣言

```
struct callinfo2{
   char id[9];          /*  利用者の名前(英数8文字) */
   int history[31];     /*  日ごとの通話時間 */
   int total;           /*  通話時間の合計 */
};
```

- 初期化

```
struct callinfo2 c1 = {"abcd1234", {0}, 0};
```

- 代入

```
char id[9] = "xyz98765";

strcpy(c1.id, id); /* 文字列 id を c1.id にコピー*/
```

- 比較

```
if (strcmp(c1.id, id) == 0){
   c1.id と id が等しい時の処理
}
```

<string.h>を
インクルードする

構造体の配列

- 実際のプログラムでは，多数の構造体を使うことが多いため，構造体を要素とする配列がよく用いられる

```
struct 構造体タグ　配列名[配列の大きさ];
```

（例）

- 定義

```
struct point table[100];
```

- 参照と代入

```
sum += table[i].x;
table[i].x = (table[i-1].x + table[i+1].x)/2;
```

【課題5-1】

構造体型pointの次の2次元配列を使って，入力した点がどの象限に含まれるかを数えるプログラムを作成せよ．ただし，入力データの最大個数は100とする．座標軸上の点はどの象限にも属さないとして，table[0]に格納するものとする．また，下のような実行例を想定する．

```
struct point table[5][100];
```

実行例

```
% ./a.exe < point-data.txt
The number of I quadrant (RUQ)    1
( 5,  1)
The number of II quadrant (LUQ)    1
(-6,  2)
The number of III quadrant (LLQ)  2
(-7, -3)
(-8, -3)
The number of IV quadrant (RLQ)   3
( 1, -4)
( 9, -4)
( 8, -4)
```

point-data.txt の内容例

```
  5   1
  0  91
 -6   2
-92   0
 -7  -3
  0 -93
 94   0
  1  -4
  9  -4
 -8  -3
  8  -4
```

構造体のtypedef宣言

```
typedef struct 構造体タグ 新しい型名;
```

- 構造体型pointに対応するPOINTという型の宣言

```
struct point{
    int x; /* x座標 */
    int y; /* y座標 */
};

typedef struct point POINT;
```

C++ではtypedef不要
左の例の通り定義すると
point num = {1, 2};
のように使える

```
typedef struct point{
    int x; /* x座標 */
    int y; /* y座標 */
}POINT;
```

構造体型の宣言と
typedef宣言を
まとめた書き方

- POINT型の変数p1の定義

```
POINT p1;
```

変数p1はPOINT型の変数と呼ばれる

構造体の引数

- 例題5-1の原点から点p1までの距離を計算する処理をdistance関数として定義
  - 構造体型pointを引数に指定してdistance関数を宣言

```
double distance(struct point p1);
```

いちいち構造体型pointを使うと関数の宣言が長くなる

  - typedef名のPOINT型を使ってdistance関数を宣言

```
double distance(POINT p1);
```

  - distance関数の定義

```
double distance(POINT p1){
   return sqrt(p1.x * p1.x + p1.y * p1.y);
}
```

  - distance関数の呼び出し

```
POINT pt1 = {2, 4};
double d;

d = distance(pt1);
```

実引数にPOINT型の構造体を指定する

【課題5-2】

- 例題5-1のプログラムでは，main関数で処理をしていた．前ページのdistance関数を用いて，例題5-1のプログラムを書き換え，動作を確認せよ．

レポート課題

- 課題5-1，5-2を実施しレポート課題として提出すること．
- 提出期限：2021年05月27日 09:00
- CLEで提出する際のファイル名（半角英数）は以下の通りとする．
  - 課題5-1の場合，XXXX-kadai5-1.c
  - 課題5-2の場合，XXXX-kadai5-2.c
  - XXXXの部分は学籍番号下4桁である．

質問について

◇質問用メールアドレス

csp-query@pn.comm.eng.osaka-u.ac.jp

◇TAへの質問について：

- まず自分で調べること．エラーメッセージを検索すると情報が得られることが多い．
- 質問を明確にすること．

  単に「わかりません…」「できません…」「おかしいです…」という質問には答えません．
  できるだけ具体的に状況を書くこと．
  エラーメッセージを記載すると回答のヒントになる．
- プログラムリストの代筆はしない．
- バグ取りはしない．