プログラミング言語演習IV
10回目　講義編

2024年 6月 13日

前橋工科大学　生命情報学科
中村 建介

今日の内容

- 構造体について
- ヘッダファイルについて

デバッグについて、

gdbのインストールについて

- sudo apt-get update
- sudo apt-get -y install gdb

どこで止まっているかを調べる
gdb：デバッガを使う

DEBUG.cというソースファイルがあったとする

```
% gcc DEBUG.c -o DEBUG
DEBUG.c: In function ‘main’:
DEBUG.c:56: warning: format ‘%d’ expects type ‘int *’, but argument 5 has type ‘int’
DEBUG.c:56: warning: format ‘%d’ expects type ‘int *’, but argument 5 has type ‘int’
% DEBUG cities.dat out
Input File Name is cities.dat
Output File Name is out
City            Prefecture      Population      Area(km^2)      Population density
Segmentation fault
%
```

```
% gcc -g DEBUG.c -o DEBUG
DEBUG.c: In function ‘main’:
DEBUG.c:56: warning: format ‘%d’ expects type ‘int *’, but argument 5 has type ‘int’
DEBUG.c:56: warning: format ‘%d’ expects type ‘int *’, but argument 5 has type ‘int’
% gdb DEBUG
GNU gdb 6.3.50-20050815 (Apple version gdb-1820) (Sat Jun 16 02:40:11 UTC 2012)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin"...Reading symbols for shared libraries ..
done

(gdb) run cities.dat out
Starting program: /Users/kenske/Desktop/2013Lecture/PE2013/04May10/SubmittedPrograms/001/DEBU
cities.dat out
setenv: Too many arguments.
Reading symbols for shared libraries +.............................. done
Input File Name is cities.dat
Output File Name is out
City            Prefecture      Population      Area(km^2)      Population density

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x0000000000004000
0x00007fff82a31422 in __svfscanf_l ()
(gdb) bt
#0  0x00007fff82a31422 in __svfscanf_l ()
#1  0x00007fff829ed17d in vsscanf_l ()
#2  0x00007fff82a2f6a8 in sscanf ()
#3  0x00000001000018c6 in main (argc=3, argv=0x7fff5fbff580) at DEBUG.c:54
(gdb)quit
The program is running.  Exit anyway? (y or n) y
%
```

コンパイル時に -g オプション

gdbを起動（プログラム名を指定）

プログラムを実行（runのあとにアーギュメントを並べる）

バックトラック

print 変数名でその時の
変数の値が表示される
うまく行かないこともある

問題のある箇所が表示される

quitと打ってgdbを停止

yと打つ

構造体について

構造体を使ったデータの整理

	人口	面積	市
GUNMA	1,974,555	6,362	12
TOCHIGI	1,979,039	6,408	12
YAMANASHI	838,260	4,465	13

構造体を使ったデータの整理

	人口	面積	市
GUNMA	1,974,555	6,362	12
TOCHIGI	1,979,039	6,408	12
YAMANASHI	838,260	4,465	13

配列だけで処理

```
char  prefecture[3][12];
int  population[3];
float  area[3];
int  num_city[3];
```

構造体を使ったデータの整理

	人口	面積	市
GUNMA	1,974,555	6,362	12
TOCHIGI	1,979,039	6,408	12
YAMANASHI	838,260	4,465	13

構造体を定義

```
typedef  struct  prefecture_info {
    char  name[12];
    int  population;
    float  area;
    int  num_city;
    }  prefecture_info;
```

```
typedef  struct  prefecture_info {
   char  name[12];
    int  population;
    float  area;
    int  num_city;
    }  prefecture_info;
```

```
prefecture_info  prefs[3];
```

```
prefs[0].population =1974555
```

	人口	面積	市
GUNMA	1,974,555	6,362	12
TOCHIGI	1,979,039	6,408	12
YAMANASHI	838,260	4,465	13

prefecture_info　prefs[3];

とする代わりに

prefecture_info　＊prefs;

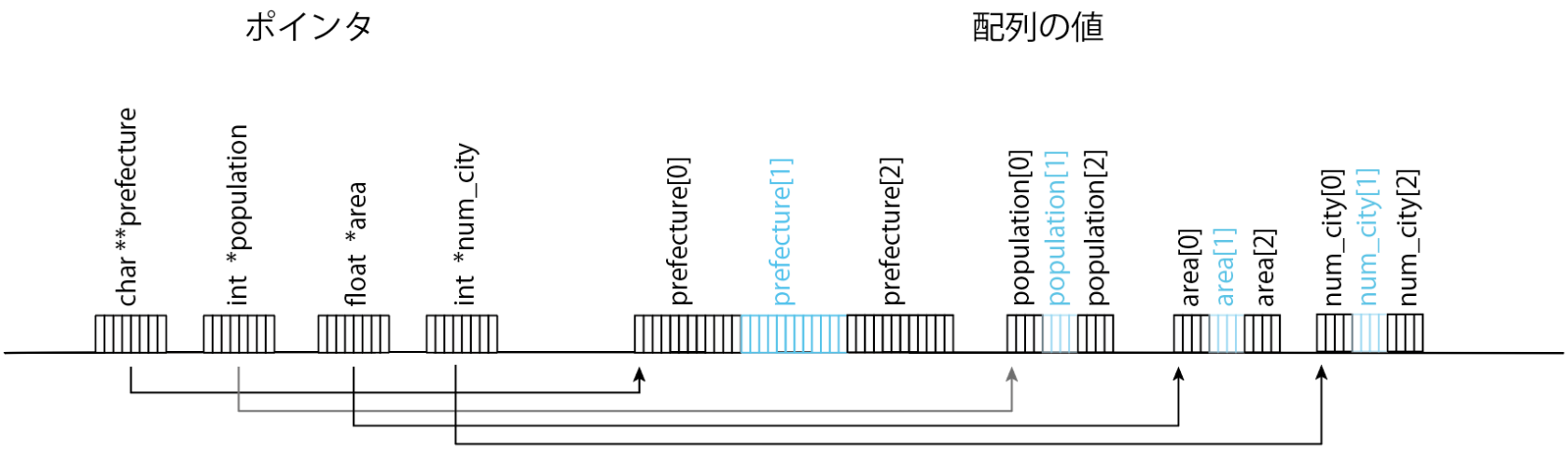
ポインタ宣言し

int　num_prefeture;

をカウントし

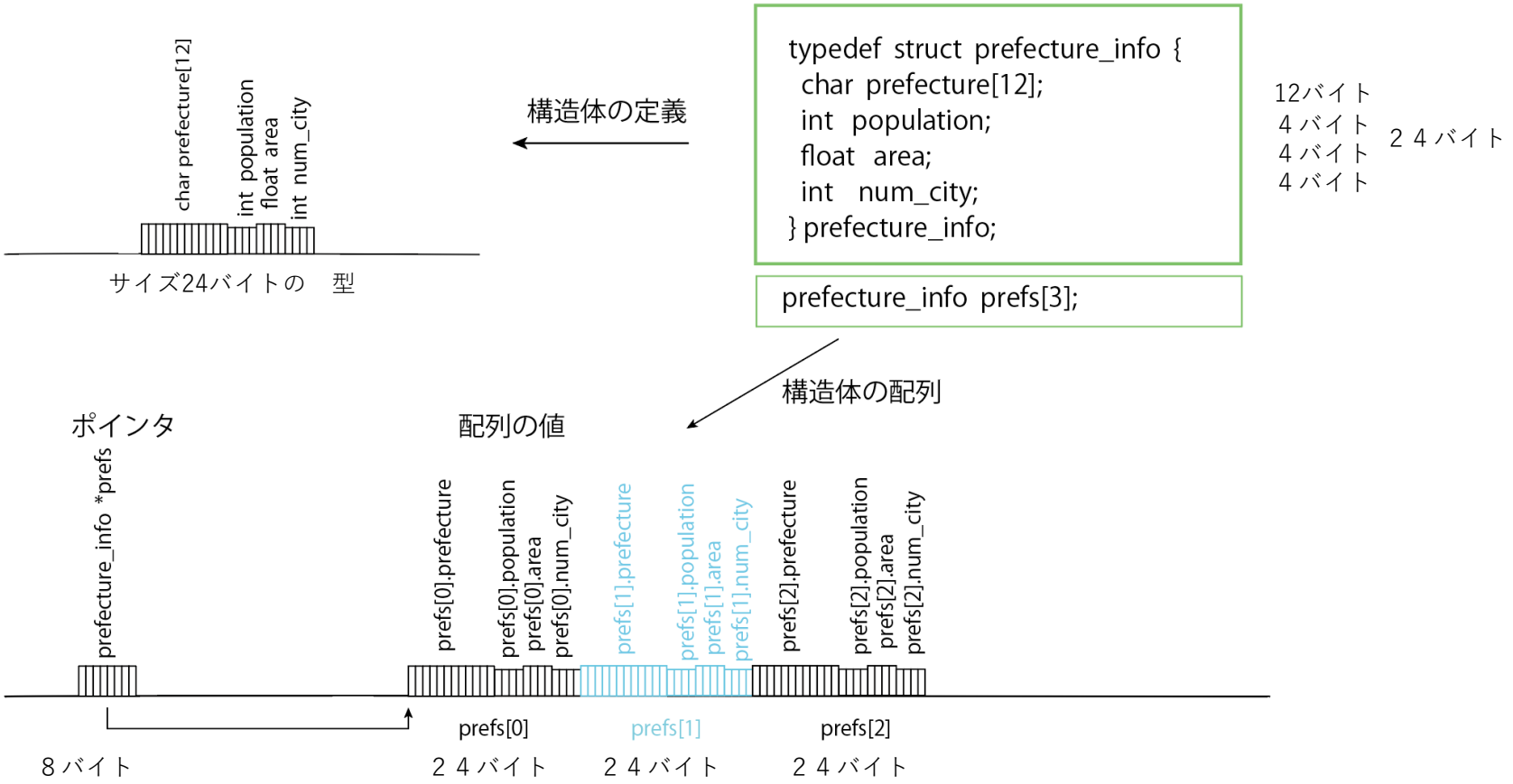
```
prefs = (prefecture_info *)malloc
        (sizeof(prefecture_info) * num_prefecture;
```

メモリ上のイメージ：別々の配列を取る方法




```
char prefecture[3][12];
        int population[3];
              float  area[3];
                    int  num_city[3]
```

メモリ上のイメージ：構造体配列



ヘッダファイル

cities.h

```
----------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_CITY   100

typedef struct city_info
 {
 char  city_name[20];
 int   population;
 float area;
 float population_density;
 } city_info;
----------------------------
```

この中身が

↓

ここに展開される

cities.cというCのソースコードを記述するファイルを作る。

cities.c

```
----------------------------
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cities.h"                    // 先に定義したヘッダファイルをインクルード（同じディレクトリ内に有る.hファイル）

int main(int argc, char *argv[])
{
city_info cities[MAX_CITY];            // 最大の想定値のサイズで構造体の配列を定義、余力があればポインタ宣言をし
                                       // 入力ファイルの行数をカウントして、動的に配列のサイズを決定してみよう

////////////////////////////////////////////////////////////////////////////////////////////////
///////////////ここに入出力ファイル名の読み込み、ファイルを開き読み込み、データを処理、出力するコードを書く///
////////////////////////////////////////////////////////////////////////////////////////////////
}
----------------------------
```

ヘッダファイルに含めるもの

- 構造体の定義
- 関数のプロトタイプ
- マクロ変数

複数のソースコード(.c）で共有する情報
ヘッダファイル中の記述１箇所だけを変更すれば
すべてのソースコードに反映される