



ce-11. 中間まとめ2

(Cプログラミング応用, 全14回)

<https://www.kkaneko.jp/cc/c/index.html>

金子邦彦



ソフトウェア開発の流れ



機能設計



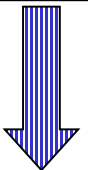
構成設計



詳細設計



論理試験



耐性試験

- ・外部仕様（プログラムの入力と出力の取り決め）

- ・内部データ構造や関数呼び出し方法などに関する取り決め

- ・ソースプログラムの記述

正しい入力データから正しい結果が得られる

- ・かテスト
関数単位からテストをおこなう

異常な入力データに対して、異常を検出でき

- ・るかテスト
異常終了することはないかテスト

- 外部仕様（プログラムの入力と出力の取り決め）



「what」を定める（「how」とは段階を分ける）

機能設計の例



入力

下記に定める「名簿ファイル」を入力とする

1. テキストファイル形式
2. 氏名, 生年月日, 住所, 電話番号が並び, 半角の空白文字で区切られる.
 - 氏名, 住所, 電話番号は最大で100バイトとする.
 - 生年月日は, 西暦年, 月, 日が「/」で区切られている
3. ファイル名は z:¥Address.txt

例

```
金子邦彦 1200/01/01 福岡市東区箱崎3丁目 392-123-8234  
○○×× 1300/12/31 福岡市東区貝塚団地 492-252-7188  
●●■ 0800/05/31 福岡市東区香椎浜1丁目 592-824-7144
```

出力

- 「名簿ファイル」の中身を, 氏名でソートして表示
 1. 氏名の順序は, c の文字列比較関数 strcmp() での順序に従う

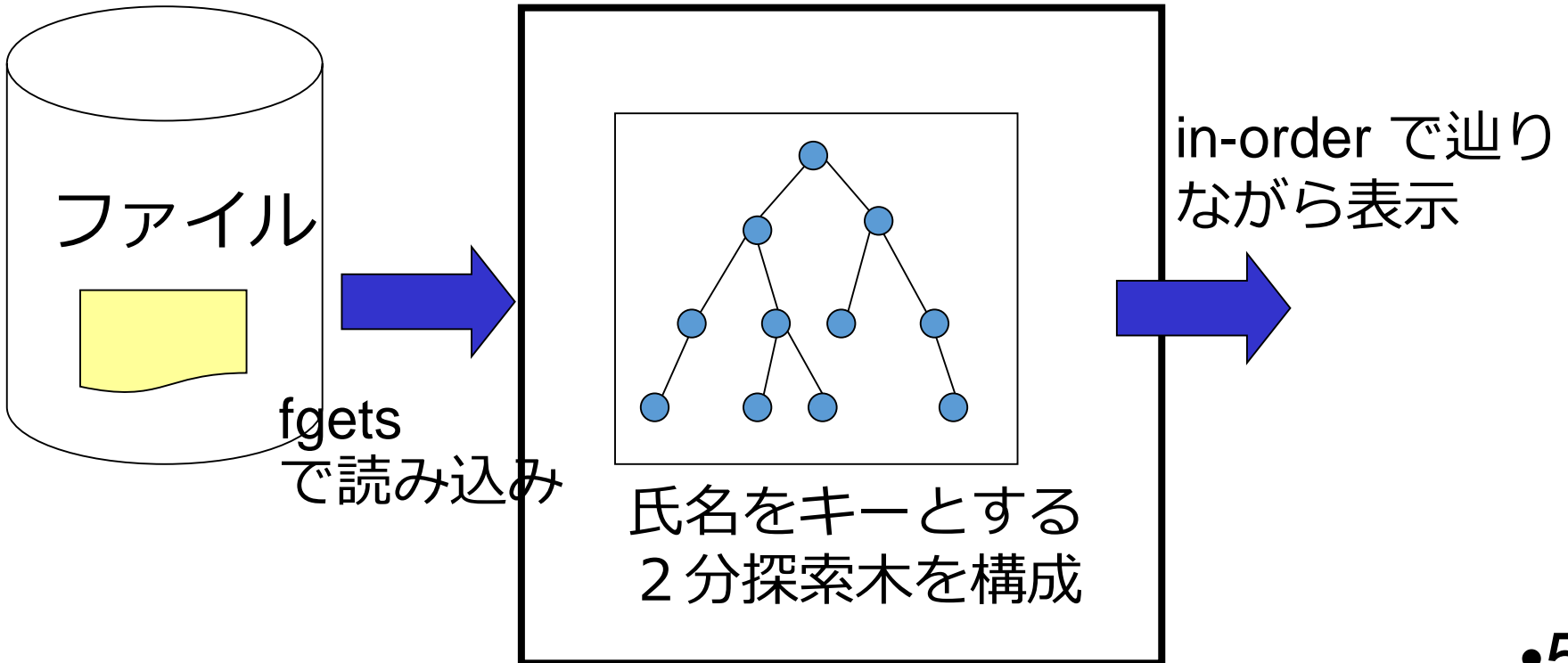
構成設計



- プログラムの内部仕様を定める
 - 外部仕様を実現するのに最も適した手段を定める

内部仕様の概要の例

プログラムのメモリ空間





構造体 **Person** の定義 (説明は後述)

構造体 **BTNode** の定義 (説明は後述)

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#pragma warning(disable:4996)
struct Person
{
    char name[100];
    int birth_year;
    int birth_month;
    int birth_day;
    char address[100];
    char phone[100];
};
struct BTNode
{
    BTNode *left;
    BTNode *right;
    Person person;
};
```

```
void print_person_data( struct BTNode *root )
{
    if ( root->left != NULL ) {
        print_person_data( root->left );
    }
    printf( "%s, %t%d/%d/%d, %t%s, %t%s\n", root->person.name, root->person.birth_year, root->person.birth_month, root->person.birth_day, root->person.address, root->person.phone );
    if ( root->right != NULL ) {
        print_person_data( root->right );
    }
}
```

print_person_data 関数

```
struct BTNode *new_person_node(Person *p, struct BTNode *y, struct BTNode *z)
{
    struct BTNode *w = new BTNode();
    strcpy_s( w->person.name, p->name );
    w->person.birth_year = p->birth_year;
    w->person.birth_month = p->birth_month;
    w->person.birth_day = p->birth_day;
    strcpy_s( w->person.address, p->address );
    strcpy_s( w->person.phone, p->phone );
    w->left = y;
    w->right = z;
    return w;
}
```

new_person_node 関数

```
struct BTNode *insert_person_node(struct BTNode *node, Person *p)
{
    if ( node == NULL ) {
        return new_person_node(p, NULL, NULL);
    }
    else if ( strcmp( p->name, node->person.name ) < 0 ) {
        node->left = insert_person_node(node->left, p);
        return node;
    }
    else if ( strcmp( p->name, node->person.name ) > 0 ) {
        node->right = insert_person_node(node->right, p);
        return node;
    }
    else {
        return node;
    }
}
```

insert_person_node 関数

```
BTNode* read_file_and_create_tree( char* file_name )
{
    FILE *in_file;
    char line[sizeof(Person)];
    Person p;
    BTNode *root;
    in_file = fopen(file_name, "r");
    if ( in_file == NULL ) {
        return 0;
    }
    root = NULL;
    while ( fgets( line, sizeof(Person), in_file ) != NULL ) {
        sscanf_s( line, "%s %d/%d/%d %s %s", &(p.name), &(p.birth_year), &(p.birth_month), &(p.birth_day), &(p.address), &(p.phone) );
        if ( root == NULL ) {
            root = new_person_node( &p, NULL, NULL );
        }
        else {
            insert_person_node( root, &p );
        }
    }
    fclose(in_file);
    return root;
}
```

read_file_and_create_tree 関数

```
int main()
{
    BTNode *root;
    int ch;
    root = read_file_and_create_tree( "z:¥¥Address.txt" );
    print_person_data( root );
    printf( "Enter キーを1,2回押してください。プログラムを終了します\n");
    ch = getchar();
    ch = getchar();
    return 0;
}
```



実行結果の例

```
仮名一郎,      1988/8/8,      福岡市東区貝塚団地,      492-252-7188
仮名三郎,      1925/12/18,   福岡市東区千早3丁目,    122-782-7244
仮名次郎,      1977/7/7,     福岡市東区香椎浜1丁目,   592-824-7144
金子邦彦,      1967/10/10,  福岡市東区箱崎3丁目,    392-123-8234
匿名次郎,      1974/8/12,   福岡市東区貝塚1丁目,    675-184-9012
匿名太郎,      1968/2/26,   福岡市東区名島4丁目,    524-892-9245
Enter キーを1,2回押してください。プログラムを終了します
```

構造体の例



char の配列 (100バイト)

int

int

int

char の配列 (100バイト)

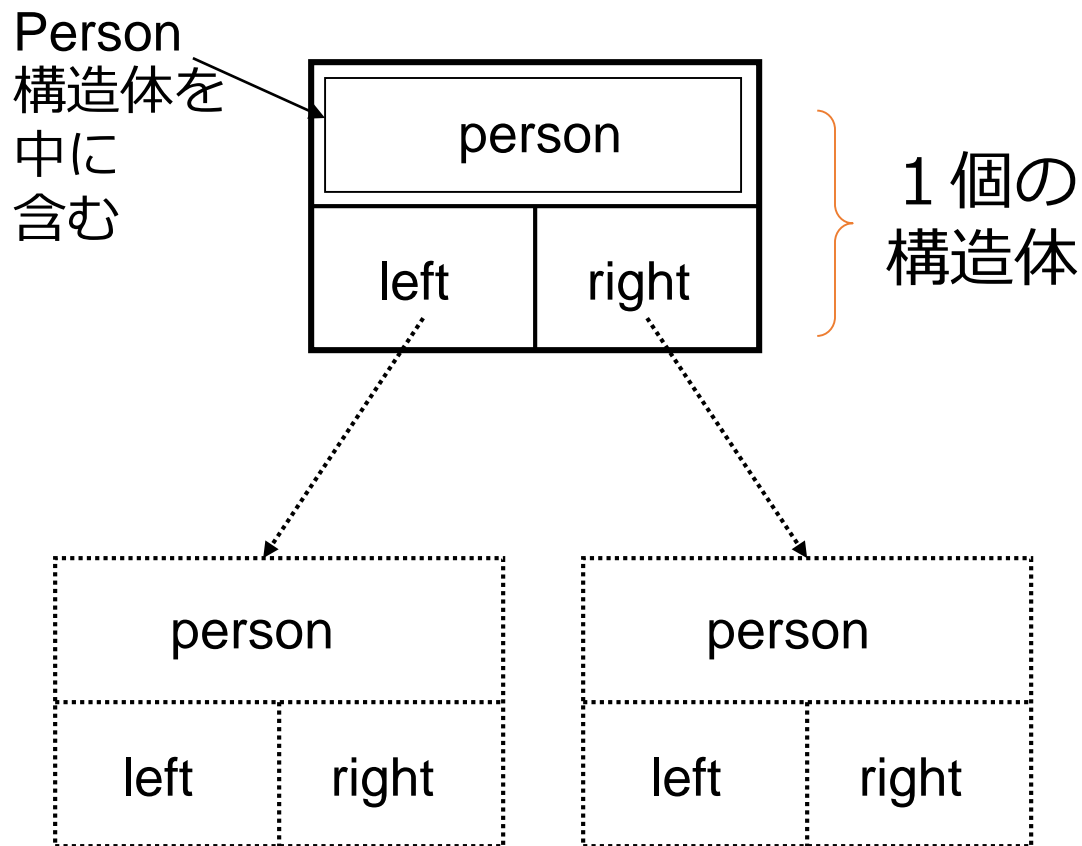
char の配列 (100バイト)

name
birth_year
birth_month
birth_day
address
phone

これで
1つの
データ

例題 1 の
構造体 Person

二分探索木の各ノードを C言語の構造体で表現



```
struct BTreeNode  
{  
    BTreeNode *left;  
    BTreeNode *right;  
    Person person;  
};
```

「Person person」の部分は、格
すべきデータに応じて変わる

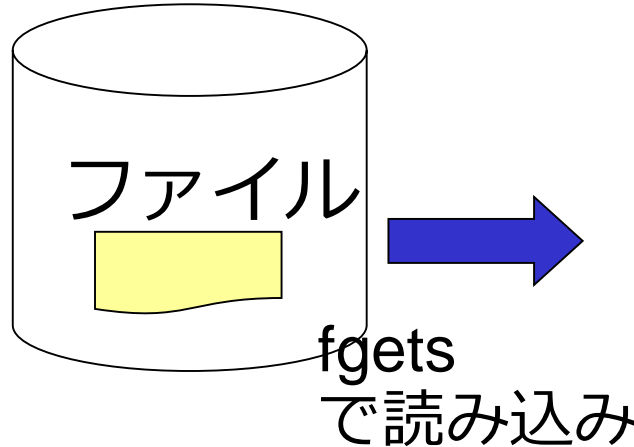


BTNode* read_file_and_create_tree(char* file_name)

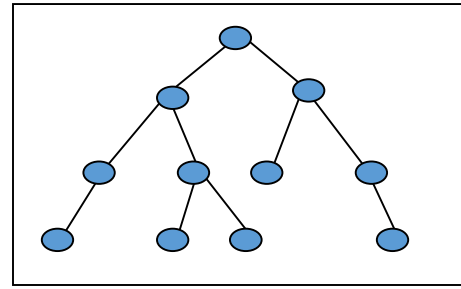
```
{  
FILE *in_file;  
char line[sizeof(Person)];  
Person p;  
BTNode *root;  
in_file = fopen(file_name, "r");  
if ( in_file == NULL ) {  
    return 0;  
}  
root = NULL;  
while( fgets( line, sizeof(Person), in_file ) != NULL ) {  
    sscanf_s( line, "%s %d/%d/%d %s %s", &(p.name), &(p.birth_year),  
              &(p.birth_month), &(p.birth_day), &(p.address), &(p.phone) );  
    if ( root == NULL ) {  
        root = new_person_node( &p, NULL, NULL );  
    }  
    else {  
        insert_person_node( root, &p );  
    }  
}  
fclose(in_file);  
return root;  
}
```

ファイルオープンに失敗した
ときのみ実行される部分

ファイルオープンに失敗したら、
プログラムが終わる



プログラムのメモリ空間



氏名をキーとする
二分探索木を構成

挿入を行う関数



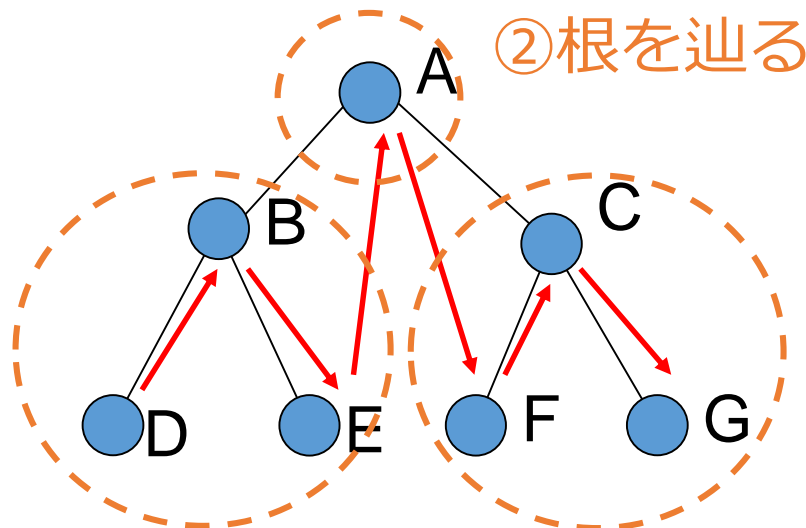
```
struct BTreeNode *insert_person_node(struct BTreeNode *node, Person *p)
{
    if ( node == NULL ) {
        return new_person_node(p, NULL, NULL);
    }
    else if ( strcmp( p->name, node->person.name ) < 0 ) {
        node->left = insert_person_node(node->left, p);
        return node;
    }
    else if ( strcmp( p->name, node->person.name ) > 0 ) {
        node->right = insert_person_node(node->right, p);
        return node;
    }
    else {
        return node;
    }
}
```

p->name . . . 挿入データの name フィールド
node->person.name . . . 探索中の部分木の根
にある name フィールド
strcmp で「辞書順」での比較



通りがけ順 (in-order traversal)

- 左の子節点以下を処理 (左部分木を辿る)
- 親節点について処理 (根を辿る)
- 右の子節点以下を処理 (右部分木を辿る)



D, B, E, A, F, C, G
の順に処理を行う

①左部分木を辿る

③右部分木を辿る

表示を行う関数 (in-order での表示)



```
void print_person_data( struct BTreeNode *root )
```

```
{  
    if ( root->left != NULL ) {  
        print_person_data( root->left );  
    }
```

左部分木

```
    printf( "%s, %t%d/%d/%d, %t%s, %t%s\n", root->person.name, root->person.birth_year, root->person.birth_month, root->person.birth_day, root->person.address, root->person.phone );
```

根

```
    if ( root->right != NULL ) {  
        print_person_data( root->right );  
    }  
}
```

右部分木