プログラミング言語論 #09

Scope

2018-06-18

C， C++を例にとってスコープについて説明する

変数の分類

```
1  int x = 0;
2  int f (int);
3
4  int main(void) {
5    int l1 = 0;
6    f(2);
7    return 0;
8  }
9
10 int f (int x) {
11   int l1 = x + 1;
12   return l1;
13 }
```

- グローバル変数
- ローカル変数
- 関数 – これは変数？

プログラム言語論専門用語

12

identifier, 識別子

プログラマが定義する名前全てを指す．変数や関数，型名，構造体名などを全てを含む．例えばそれが変数なのか関数なのかは区別しない．

scope, スコープ

識別子が使える（未定義エラーにならない）空間的範囲．

空間的とは，このファイルの X 行目から Y 行目というように指定できること

時間的な範囲（この関数を実行中に限り〜など）を指す用語もある (extent) が，省略．

スコープに基づく C の変数，関数の分類

- （開始点はともかく）ファイル全体と一致するスコープ
  - グローバル変数，グローバル関数 → グローバルな識別子
  - 他のファイルでも使いたいなら extern 宣言する
- ある関数に対応するスコープ
  - ローカル変数（引数はこの中に含まれる；ローカル関数は言語仕様にはない）
- 関数よりももっと小さな範囲に対応するスコープ
  - これもローカル変数という

```
1  int main (void) {
2   ...
3    if ( ... ) {
4      int tmp = ...;
5      ...
6    }
7    ... }
```

プログラミング言語論専門用語：再定義

13

block, ブロック

複数の文をまとめたもの． C では{と}とで囲まれる．

ブロックの例

- 関数定義は 1 つのブロック
- if 文の条件式の後
- while 文の条件式の後

```
1  int main (void) {
2    int x = 0;
3    if (x < 0)
4    {
5      int y = - x * 10;
6      ...    }
```

local, ローカル

そのスコープが（ファイルより小さな）ブロックに対応すること

global, グローバル

そのスコープがファイルに対応すること

スコープに基づくC++の変数の分類

- （開始点はともかく）ファイル全体と一致するスコープ
  - グローバル変数
  - 他ファイルに輸出するには extern 宣言か**名前空間**を使う
- ファイルより小さなブロックに対応するスコープ
  - ローカル変数（C と同じ）
- クラスの中で定義されている変数
  - インスタンス変数
  - そのスコープはスコープ修飾子で個別設定できる

```
1  class C {
2  ???:          // ???がスコープ修飾子，次のが出現するまで影響下
3    int x;      // xがインスタンス変数
4    void f1() { ???によらずxが使える };
5    void f2() { ???によらずxが使える };
6  };
7
8  int main (void) { xが使えるかどうかは???次第 }
```

C++, Java での代表的なスコープ修飾子

アクセス修飾子ともいう

public

どこからでも使える．グローバルと同じ．

private

そのクラス（の関数）でのみ使える．複数の関数にまたがっているので関数をスコープとするローカル変数よりは広いスコープを持つ．

protected

そのクラスと，そのクラスに包含されるクラス（の関数）でのみ使える．

見れば分かるように main 関数はどのクラスにも，

public, protected によるスコープ指定の例

```
1  class C1 {
2  protected: int x, y; // Cでの構造体のフィールド変数に相当
3  public:    int result (void) { return x + y; }
4             int reset (void) { ... }
5  };
6
7  class C2: public C1 ...  // C1に包含される別のクラスC2の定義
8
9  int main(void)
10 {
11   C1 a(1,1);  // インスタンス a
12   C1 b(1,1);  // インスタンス b
13   C2 c(3,4);  // インスタンス c
14   a.reset();  // できる
15   a.x + b.y;  // できない
```

- クラス C1 のインスタンス a はインスタンス変数 x, y を持っている
- クラス C1 の別のインスタンス b は個別に x, y を持っている
- C1 に包含される C2 クラスのインスタンス c も独自の x, y を持っている
- x,y は関数（メソッド） result, reset から使える
- しかし関数 main からは使えない
- これらはスコープ修飾子 protected によってスコープが限定されている

インスタンス変数のスコープの選択問題

ボタンそのものはグローバル変数だとする．その座標はグローバル変数にする必要があるか．

- ∞：グローバル変数：通常はセキュリティ的に問題（main 関数で勝手にリセットできるなど）
- 1：ある関数のローカル変数：関数が終わると座標がなくなる？
- ボタンに関する関数群で共有：もっとも適切なスコープ

1 と∞はダメ．これは型制約を導入したのと同じ構図：

- 型全部はダメ
- 型 1 つはダメ
- 特定の型群が適切