プログラミング演習
～ガイダンス～

1　目的

本演習は，C 言語によるプログラミングスキルの習得である．言語によらず”プログラミング”とは，”道具”である．人間がコンピュータに何か仕事をさせたいときに，プログラミングという”道具”を用いて，コンピュータによる”処理”を作成する．このようなスキル (技術・技能) を効率よく身につけるためには，座学と実践のよいバランスに加え，具体的な作製物である目的・目標が重要である．

例えば，トンカチやノコギリ，カンナといった道具を考える．単機能であり単純な道具であるために，それほど座学は必要ではないが，トンカチに付属の釘抜きやカンナの調整方法などは，教えてもらった方が効率がよい．口伝だけではなく，道具を扱うスキルは，実際に扱って身につけることが重要になってくる．トンカチであれば釘をうち，ノコギリであれば木を切ってみる．その背景には，材木でなにかを作りたい (家や家具，ちょっとした台など) が意識下・無意識下にあるために，反復練習が有用であるのだが，もし”目的・目標”がないとしたらどうであろう? 物を作る，という意識なしの，トンカチの上下運動，ノコギリやカンナの引いたり押したり，では，反復練習を通して得られるものは少ない．

プログラミングスキルの習得において難しいことは，この作るものをイメージしにくい，ということが上げられる．それはコンピュータが他の道具に比べて非常に高機能かつ汎用的な道具であるのに加え，”作成したものが見えにくい”もしくは”作った過程と作られたものが直接結びついて見えない”からであると思われる．

そこで，演習を通して共通の目的・目標を掲げ，それとなるべく関係するように演習 (反復練習) を行うことで，”使えるプログラミングスキル”の取得を目指している．

ここで共通の目的・目標として，N 本腕バンディット問題を例題にする．最終目的・目標としては，この N 本腕バンディット問題を自動的に解くプログラムの作成である．次に N 本腕バンディットの説明，ルール，注意点などを述べる．

2　N 本腕バンディット問題

N 本腕バンディットとは，N 本のレバー (以下，腕と呼ぶ) を持つスロットマシンのようなゲームである．N 本腕バンディットは，N 本の腕を持つ．各腕には当たり確率が設定されており，ゲームプレーヤーは 1 つの腕を選択することで”当たり”または”はずれ”が決定する．”当たり”となると，プレーヤーは報酬 (お金など) を得ることができる．報酬の量も腕によって変えることができる．また，一回の腕の選択を 1 試行 (1 プレイ) 呼ぶ．このようにスロットマシンではボタンを使用するが，バンディットではゲームプレーヤーは腕の選択のみを行う．そして N 本腕バンディット

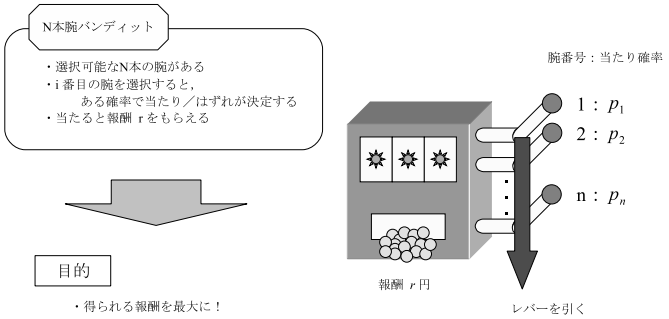

図 1: N 本腕バンディット

問題とは，N 本の腕をもつバンディットに対してどのようにすると多くの報酬を得ることができるか，を解く問題となる (図 1)．特に，プレーヤー間の競争が存在する場合，他のプレーヤーよりも多くの報酬を得ることが目標となる．

3 N 本腕バンディットにおけるプレーヤーのスコア

N 本腕バンディット問題において，他のプレーヤーとの成績比較のためにスコアがある．スコアとは，そのプレーヤーが得た報酬である．値としては同じであるが，

- バンディットから得られるものが報酬
- プレーヤーの成績として表されるものがスコア

である．

特によく使用する値として

- 報酬 = スコア: 基本的に 1 回毎の成績
- 総報酬 = 総スコア: 10 回や 20 回など決まった数だけバンディットを試行 (プレイ) した場合の総合成績
- 平均報酬 = 平均スコア: 複数回バンディットを試行 (プレイ) した場合の 1 回あたりの成績

である．

3.1 N 本腕バンディットを経験してみよう

3.1.1 とりあえず実行

- はたおり虫より gameplay**(**は 00,01,02 のような数字) をダウンロード
- kterm 上で実行
  - chmod +x gameplay**
  - ./gameplay**

3.1.2 コンパイルして gameplay 作成

- はたおり虫より以下のファイルをダウンロード
  - bandit.h
  - player.c
  - bandit00.o 〜bandit08.o
- 以下を実行してコンパイル (bandit00.o を対象．bandit01.o〜bandit08.o も同様)
  - gcc -o gameplay00 player.c bandit00.o
- 作成した gameplay00 を実行
  - ./gameplay00

3.2 演習前半部

N 本腕バンディットを例題として，プログラムの各機能の演習を行っていく．ここで，例題として N 本腕バンディットを扱うので，関連する情報を以下に記載しておく．現在は分からなくてもよいので，各演習時において参考にして，必要に応じて質問するように．

3.2.1 N 本腕バンディットプログラムの実体

N 本腕バンディットに関するプログラムは，はたおり虫を通して配布する．配布ファイルの形式は，N 本腕バンディットに関するヘッダーファイルと実体であるオブジェクトファイルである．ここで，**は 00 から 99 までの識別番号である．スロットマシンに色々な設定があるように，バンディットも色々と設定を変えることができる．ここでは選択できる腕の数や当たり確率，報酬の量を変えており，違う識別番号のバンディットでは違う設定となっている．

- ヘッダーファイル名: bandit.h
- オブジェクトファイル名:bandit**.o

3.2.2 N 本腕バンディットのプログラム使用方法

識別番号によってバンディットの性格は異なるが，使い方は同じである．プログラム中でバンディットを使う場合には，下記の関数を使うことになる．次回の演習 (プログラミング演習〜基礎文法〜[1]) にて実際に使用する演習がある．

- ヘッダーファイル名: bandit.h
- オブジェクトファイル名:bandit**.o
- 関数
  - void init_bandit(void)
    - バンディットの初期化．最初に一回使用する必要あり．

- double bandit(int arm)
  - バンディット 1 試行
  - 引数:arm:選択する腕．N 本腕の場合には，1〜N の数.
  - 返り値:報酬は正の値．はずれている場合は 0.0 となる．選択した腕 (arm) がおかしい場合は-1.0 となる.
- int get_arm_num(void)
  - 選択可能な腕の数を知るための数．必要があれば使用.
  - 返り値:選択可能な腕の数.

3.2.3 N 本腕バンディット使用上の注意

※分からなければ特に問題なし.

乱数を使用する場合，srand() と rand() を使用する．ここで，srand() は乱数を使用するための初期化処理であり，プログラム中に一回のみの実行となる．一方，rand() は乱数を発生する関数であり，乱数が欲しい場合には毎回呼び出す．ここで，srand は init_bandit() 関数の中で実行しているので，プログラム中では 絶対に使用しないこと.

3.3 演習後半部 (時間があれば)

演習で習得したスキルを用いて，N 本腕バンディット問題を自動的に解くプログラムの作成を行う.

N 本腕バンディットに対して，プレーヤーはより多くの報酬を得ることを目標とする．本演習では，プレーヤーは人ではなく N 本腕バンディットを解くプログラムである．ここで，成績の評価方法を統一することで，各プログラム作成者がつくった自作のプレーヤーの”よさ”を比較することができる．または各プログラム作成者のアイデアの”よさ”の比較ができる．そこで，以下のようにスコアの計算方法を定義する (図 2).

- プレーヤーが N 本腕バンディットの腕を 1 回選択する行為を”1 試行”とする.
- 最初の n 回はバンディットの特性をみるためのテスト試行で公式スコアの計算には入れない
- n 回後のテスト試行後，連続した t 試行を行い，得た報酬の合計 (総スコア) をプレーヤーの成績とする.

テスト試行回数 n 回および本番試行 t 回は演習中に指示する.

ちなみに，スコアは本演習の成績には 全く考慮しない．これは本演習の目的はあくまでプログラミングスキルの取得であり，バンディット問題を解くことではないからである.

3.3.1 公平なスコア計算の実現

スコアの計算法方が人によって異ならず他の人と同じ方法で計算することは，公平な教義では重要である．ここでは，公平なスコア計算を行うために以下のことを考える.

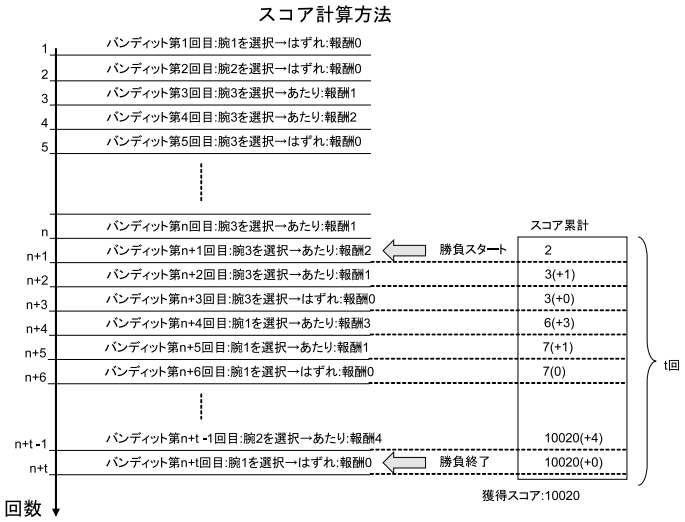

図 2: スコア計算方法

- プログラム作成者は，プレーヤー (N 本腕バンディットを解くプログラム) を以下の用に関数化する．
  - void set_arm_num(int arm_num);
    - バンディットの腕の数 (arm_num) をプレーヤーが知るための関数
  - int decision_making(int previous_reward);
    - decision_making とは”意思決定”の意味であり，”バンディットのどの腕を選択するかを決める関数である．
    - 入力は，意思決定 (次にどの腕を選ぶか) を行うのに必要な情報である．
    - 入力 previous_reward は前回自分が選んだ腕に対して得られた報酬である．これを元に次の選択する腕を選ぶことを考える．
    - 出力 (返り値) が，この意思決定を行う関数で選択する腕である．
  - 上記関数を以下のファイルで実装し，オフィシャル (教員) に提供する．
    - player**.o(**は対応するバンディットの識別番号と合わせる)
      - 上記関数を player**.c として作成 (**は対応するバンディットの識別番号と合わせる)
      - gcc -c player**.c で player**.o を作成
  - オフィシャル (教員) は提供された関数をもとにスコア計算プログラムによってスコア計算を行う．

4 参考文献

参考文献

[1] プログラミング演習～基礎文法～