第1章 量子コンピュータのアプローチ①…手元で動かしてみる

C言語シミュレーションで動作原理をつかむ

藤井 啓祐

リスト1 量子ビットのC言語記述

```
double _Complex comp_amp[2];

comp_amp[0] = 1.0/sqrt(2) + 0.0i;
comp_amp[1] = 1.0/sqrt(2) + 0.0i;
```

本稿では，量子力学の原理やルールをC言語でシミュレーションしてみます．ソースコードを眺めながら量子コンピュータの動作原理を追ってみましょう．

量子計算の基本ルール

■ その①…量子ビットの記述

● 量子ビットは複素数のベクトルで表せる

量子ビットは，2つの複素数を用いて書き表せます．従来のビットに対応する0や1をこの複素ベクトルで書くと，

$$|0\rangle=\begin{pmatrix}1\\0\end{pmatrix},\quad |1\rangle=\begin{pmatrix}0\\1\end{pmatrix}$$

のようになります．0と1が重ね合わさった状態は，2つのベクトルの和として

$$\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle=\begin{pmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{pmatrix}$$

のように記述されます．

● C言語によるシミュレーション

これをC言語で表現する方法を考えましょう．上の例では，実数しか用いていませんが，一般的には複素数の値をとってもよいので，変数の型は，double _Complexを用いる必要があります．量子ビットは2つの複素数の配列で定義されるので，**リスト1**のように記述すれば量子ビットの重ね合わせ状態を1つ準備したことになります．C言語でシミュレーションする場合は，この配列を参照し，comp_amp[i]の値を取得できますが，実際の量子ビットではcomp_amp[i]に相当するものを直接出力させることはできません．

リスト2 乱数を生成するrand関数を使いビット値を確率的に得る測定

```
int measurement(double _Complex comp_amp){
  int outcome;
      ←0～1の一様乱数
  if(rand() < pow(cabs(comp_amp[0]),2)){
    outcome = 0; // 0に確定
    comp_amp[0] = 1;
    comp_amp[1] = 0;
  }else{
    outcome = 1; // 1に確定
    comp_amp[0] = 0;
    comp_amp[1] = 1;
  }
  return outcome;
}
```

実際に量子コンピュータ上の操作として許されているものは「測定」という操作だけです．量子ビットを測定すると0もしくは1に確定した状態が得られます．0もしくは1を得る確率は，複素確率振幅comp_amp[i]の絶対値の2乗として得られるので，測定は，**リスト2**のような関数によってシミュレーションできます．

リスト2に示した関数の出力としてビット値を確率的に得ることになります．また，測定後の状態は，測定結果に従い，0か1かが確定した状態になっています．つまり量子状態は，測定せずに重ね合わせのままいるときに量子的に振る舞い，0か1かをアドレスすると複素確率振幅の2乗でビットがサンプルされ，状態が0であるか1であるかが確定します．

■ その②…1量子ビットの基本演算

● 量子ビットの演算には行列を使う

量子コンピュータでは，量子演算を量子ビットに作用させて計算を行うことになります．ただし，量子力学は線型であること，複素ベクトルは規格化されていることを考慮すると，複素ベクトルに許された演算はユニタリー行列（Unitary Matrix）に限定されます．ユニタリー行列とは，転置および複素共役をとった行列が自分自身の逆行列になるような行列です．

例えば，NOT演算に対応するビットの反転を量子