16\_基本情報アルゴリズム②

**課題1**

入力文字列をランレングス法により符号化し、表示するプログラムを作成しなさい。

※ランレングス符号化とは

FAX等の画像データの圧縮に用いられる符号化方式の一種で、連続する同一の値を「色×回数」という列の長さの情報に置き換える方式のこと。

例えば「白白黒黒黒白黒黒」という画像データがあった場合、「白2黒3白1黒2」に変換できる。

単純な画像データほど圧縮率が上がり、可逆圧縮であるため完全に画像データを再現できる。

ファイル名：Kad16\_1.java

＜実行結果＞

文字列＞白白黒黒黒白黒黒白白白白黒黒白黒白白黒黒黒

$白:2 $黒:3 $白:1 $黒:2 $白:4 $黒:2 $白:1 $黒:1 $白:2 $黒:3

＜実行結果＞

文字列＞すもももももももものうち

$す:1 $も:8 $の:1 $う:1 $ち:1

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | String | runLength | String data |

◆runLengthメソッドの処理

引数で受け取った文字列を元にランレングス法を使用して出力文字を作成し、戻り値として返す。

①以下の変数を定義する。

int i = 0; // 文字列インデックス

int count = 1; // 文字数カウンタ

String outStr = ""; // 出力文字

String subStr; // 部分文字

②subStrに入力した文字列の先頭文字を抜き出し代入する。

subStr = data.substring(i, i + 1);

※substringメソッドとは

引数に（開始位置,終了位置）を指定することで、文字列から指定部分文字を抜き出した結果を返す。

③抜き出した文字（subStr）とその次の文字をequalsメソッドを使用して比較する。

次の文字が同じ場合は文字数カウンタをインクリメントする。

違う場合は出力文字を連結し、次の1文字を抜き出して文字数カウンタを初期化する。

これを先頭から末尾の1つ前の文字まで繰り返す。

※equalsメソッドとは

文字列と文字列を比較し、一致していればtrue、していなければfalseを返す。

String型はリファレンス型の為初期値以外は == で比較できない。

//入力文字列の先頭から末尾の１つ前の文字までiをなぞる

for(index = 0; index < data.length() - 1; index++){

//次の文字を抜き出し、比較する

if(subStr.equals(～省略～)){

//同じ文字が続く場合はcountをインクリメントする

count++;

}else{

//違う文字に遭遇したら繰り返し文字と回数を出力文字変数に連結する

outStr += " $" + subStr + ":" + count;

//次の文字を抜き出し、部分文字を更新する

subStr = ～省略～;

//カウンタを初期化する

count = 1;

}

}

④ループを抜けたら最後の文字の繰り返し文字と回数を出力文字代入変数に連結する。

outStr += " $" + subStr + ":" + count;

**⑤**出力文字を戻り値として返す。

◆mainメソッドの処理

①String型変数inStrに入力した文字列を代入する。

②runLengthメソッドを入力文字列を引数として呼び出し、戻り値を表示する。

**課題2**

アルファベット文字列から力任せせ法により入力文字列を探索し、発見した文字位置を表示する

プログラムを作成しなさい。

※力任せ法(Broute Force Search)とは

探索文字列の先頭から末尾まで1文字ずつ総当たりで特定の文字列を探し出すアルゴリズム。

効率は良くないが、探索アルゴリズムの中では最も単純で記述しやすい。

力任せ法の他にも優れたKMP法や、更に優れたBoyer-Moore法などが存在する。

ファイル名：Kad16\_2.java

＜実行結果＞

データ：ABCDEFGHIJKLMNOPQRSTUVWXYZ

検索文字列＞ABC

ABCは1文字目に見つかりました。

＜実行結果＞

データ：ABCDEFGHIJKLMNOPQRSTUVWXYZ

検索文字列＞E

Eは5文字目に見つかりました。

＜実行結果＞

データ：ABCDEFGHIJKLMNOPQRSTUVWXYZ

検索文字列＞XY

XYは24文字目に見つかりました。

＜実行結果＞

データ：ABCDEFGHIJKLMNOPQRSTUVWXYZ

検索文字列＞ABXY

ABXYは見つかりませんでした。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 戻り値 | 名前 | 引数 |
| メソッド | int | bruteForce | String data, String search |

◆bruteForceメソッドの処理

引数で受け取ったdata（探索対象文字列）からsearch（検索文字列）を力任せ法で探索し、

発見した文字位置を戻り値として返す。見つからなかった場合は-1を返す。

①探索対象文字列と検索文字列を1文字ずつ比較していく為のカーソルをそれぞれ宣言する。

int dc = 0; //data(探索対象文字列)をなぞるカーソル

int sc = 0; //search(検索文字列)をなぞるカーソル

②両方のカーソルがそれぞれの文字列の長さと違う（カーソル位置が末尾でない）間、

以降の③の処理をループする。

③探索対象文字列と検索文字列からそれぞれカーソル位置の1文字をcharAtメソッドを使用して

参照し、比較する。

　参照した1文字が等しい場合、カーソルを次の文字へ移動する。

　等しくない場合、探索対象文字列カーソルを検索文字と一致した数分戻し、更に次の文字に移す。

　その後、検索文字列カーソルを先頭の文字へ移動する。

　※charAtメソッドとは

　文字列.charAt(n)で指定したn番目の文字を参照し、戻り値として返します。

//探索対象文字列と検索文字列から抜き出した1文字が一致している場合

if(data.charAt(dc) == ～省略～)){

//次の文字へカーソル移動

dc++;

～省略～;

}else{ //不一致の場合

～省略～; //探索済み文字列の次の文字へカーソル移動

sc = 0; //検索文字列の先頭の文字へカーソル移動

}

④③のループ処理後、検索文字列カーソルが検索文字数と等しい場合、探索成功とみなし

発見した文字列の先頭位置+1を戻り値として返す。

　等しくない場合、探索失敗とみなし-1を返す。

//検索文字列カーソル位置が検索文字数と同じ（完全一致した）場合

if(sc == search.length()){

return ～省略～; //探索成功

}

return -1; //探索失敗

◆mainメソッドの処理

①アルファベット文字列（探索対象文字列）をString型定数で宣言する。

final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

②String型変数inStrに入力した検索文字列を代入する。

③探索対象文字列と検索文字列を引数にbruteForceメソッドを呼び出し、

戻り値をint型変数resultに代入する。

int result =　～省略～;

④力まかせ探索を行った戻り値が、-1の場合は見つからなかったメッセージを、

　それ以外の場合は何文字目に見つかったかを表示する。