【講義メモ】 担当：平野正喜（ひらのまさき）
この講座ではプロジェクタに講義メモを書きながら進めます。この文字サイズの読める席にお座りください。
18:15～20:45（途中休憩有）。受講者数は１３人です。
この講義メモは講義終了と同時に下記のサイトにPDFで掲載し、ダウンロード可能にします。ご利用ください。次回予告も掲載します。質問やコメントが送信可能です。
https://tkuip.rundog.org
前回の１問：接頭語を小さい方から大きい方へ並べたものとして正しいものは？
ア KGμP イ mKMG ウ PTGM エ npkμ 正解はイ
p（ピコ）n（ナノ）μ（マイクロ）m（ミリ）～K（キロ）M（メガ）G（ギガ）T（テラ）P（ペタ）となる。
p.155 7-3-3「文字の表現」から
・【補足】文字コード：コンピュータ内部で文字を表すための番号で、OSや用途によって複数のコード（体系）が兼用されている。Windows ではUnicodeとシフトJISコードが兼用されている。
・【補足】文字コードの進化による理解：
① ASCII：英数字のみ最大１２８文字なので2^7種類なので、１文字7ビット。
② JIS8ビット：ASCIIにｶﾅ（半角カナ）を加えて最大２５６文字にしたもので2^8種類なので、１文字8ビット。
③ JIS１６ビット：JIS8ビットの２文字分でひらがな、漢字もコードに含めたもの。「ここから漢字」「ここまで漢字」を意味する制御文字が必要で、効率が悪い
④ シフトJISコード：③の文字の配置をずらして（シフトして）制御文字を不要にしたもの。
⑤ Unicode：世界中の文字を扱えるように１文字を１６ビット以上にしたもの。扱いたい文字の広さに応じて使い分け可能。
・EUC＝エクステンデッド（拡張）UNIXコード：上記の進化とは別に、サーバなどで用いるOS「UNIX」ように作られた文字コード。１文字１６または２４ビット（＝2、3バイト）。
p.155 7-3-4 論理的な記述
・【補足】述語論理：AIなどで用いられる命題の記述法。命題の主語と述語を記号を用いて表す。
・【補足】演繹推論：普遍的な前提から、個別の事例を推論すること（当たりやすい）
・【補足】帰納推論：個別の事例から、普遍的な前提を推論すること（例外が多くなる）
p.156 7-3-5 AI（人工知能）の技術
・【補足】機械学習：コンピュータに学習機能をもたせたもの、また、その機能。３種類有る。
① 教師あり学習：正解のある質問を正解と共にAIに学習させること。例：手書きの文字画像と、それが何の文字かを与える。
② 教師なし学習：大量の情報をAIに与え、AIが自ら判断して分類できるようにすること。例：ブレーンストーミングで得た大量の情報の分類
③ 強化学習：AIの行った判断に評価点（報酬）を与え、より高い評価が得られるように、AIに試行錯誤を繰り返させること。学習精度を向上できる
・【補足】アノテーション：データや画像に説明をつけたもの。教師あり学習の教材になる
・ニューラル（神経）ネットワーク（網）：人間の神経回路を真似た処理回路。情報を次々と与えることで、網のつながりが変化し、AIとして進化する
・ディープ（深層）ラーニング（学習）：ニューラルネットワークを多層化したもの。情報に対する

対処(判断)の結果が網の目の中に蓄積されることで、AIとして進化し、予測などの高度な利用が可能。

p.158 7-4-1 データおよびデータ構造

・【補足】型(データの型):コンピュータがデータを扱うには、データを格納するための必要な記憶領域の大きさと、データの扱い方を知る必要がある。これを示す情報が(データの)型。

① 文字型:文字コードで決まる1文字の大きさを基本とし、1文字を格納できるデータの型。これを応用して0文字以上の複数文字を扱えるのが文字列型。

② 数値型:整数や実数(小数点がある値)を格納できるデータ型。格納したい値の範囲に応じる大きさになる。例:int 型=±21億程度の整数を格納できる代表的な数値型

③ 論理型:真理値(p.144)を格納できるデータ型。データの容量は1ビット。通常、真をtrue、偽をfalseと表す。条件の可否をそのまま論理型のデータとして扱える。

・【補足】データ構造:データを1個1個ばらばらに扱うのではなく、集合として扱う手法。

① レコード:データ1個を項目(フィールド)とし、項目を並べたものをレコードとして扱う手法。1つのレコードを1件ともいい、複数のレコードを格納する構造をファイルという。例:学生レコードにフィールドとして名前(文字列)、年齢(整数)、合否(論理)を持たせる。
※レコードを構成するフィールドのデータ型はバラバラでも良い

② 配列:同じ意味で同じデータ型のデータを順番に並べたもの。大量の情報を「○○の□番目」として扱える。この□を添字、指標、インデクスという。大量の情報から任意の1件を高速に得られるが、途中の挿入や途中の削除には時間がかかる。

③ リスト:同じ意味で同じデータ型のデータに「次はどこか」という情報(ポインタ)を与えたもの。大量の情報を先頭から順に扱える。実際の格納場所に依存しないので、効率が良く途中の挿入や途中の削除も簡単だが、大量の情報から任意の1件を得るには先頭から探すので遅い。改良型として「前はどこか」という情報も持つ双方向リストもある。

p.161 7-4-2 キューとスタック

・【補足】キューとスタック:配列やリストなどで実現できるデータ構造の応用方式の代表

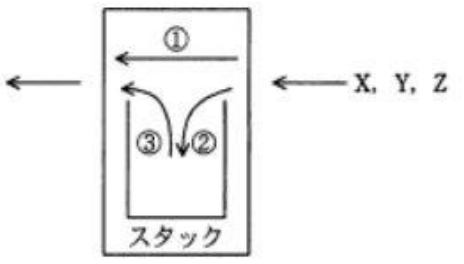
・【補足】キュー:データを格納できる構造で、取り出すと、もっとも前に格納されたデータが得られる。よって、待ち行列の構造。キューへの格納をENQ、キューからの取り出しをDEQという。

例: ENQ(❶)、ENQ(❷)、ENQ(❸) とすると、キュー内部は [❶❷❸] となる。
ここでDEQとすると、❶が取り出され、キュー内部は [❷❸] となる。
ここでDEQとすると、❷が取り出され、キュー内部は [❸] となる。

・【補足】スタック:データを格納できる構造で、取り出すと、もっとも最近格納されたデータが得られる。スタックへの格納をPUSH、スタックからの取り出しをPOPという。

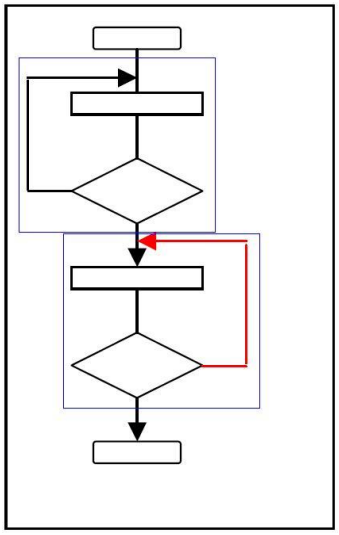
例: PUSH(❶)、PUSH(❷)、PUSH(❸) とすると、スタック内部は [❸❷❶] となる。
ここでPOPとすると、❸が取り出され、スタック内部は [❷❶] となる。
ここでPOPとすると、❷が取り出され、スタック内部は [❶] となる。

この例で分かる通り、スタックを用いるとデータの順序を変更できる(右図は出題例で、②がPUSH、③がPOP)。なお、キューとスタックを併用することで、データの複雑な操作が可能。



例: ENQ(❶)、ENQ(❷)、ENQ(❸)、PUSH(❹)、PUSH(DEQ)、POP すると❶が得られる

p.163 7-5-1（アルゴリズムとプログラミング）流れ図
・【補足】アルゴリズム：問題を解決したり処理したりする手順。料理のレシピに当たる。箇条書きで示すこともあるが、複雑な場合に記述するための図法がある。
※アルゴリズムは著作物ではないので著作権法（p.35）の対象外
・【補足】流れ図：アルゴリズムを表すための図法の1つで、国際標準。定められた記号を用いることで、他者にアルゴリズムを伝えたり、複数人で評価することが可能。
p.164 7-5-2 アルゴリズムの基本構造
・【補足】基本構造：構造化技法（p.103）と同様に、アルゴリズムは3つの構造のみで記述することが基本。
※ なお、流れ図では基本構造に従わないアルゴリズムも記述できてしまうが、他の図法では記述できない場合が多い



① 順次構造：上から順に行う構造で、内側に他の構造を持つことが多い（右図は2つの繰返し構造が順次構造になっている）
②選択構造：条件によって分岐する構造で、3つ以上に分岐する多分岐構造も可能。構造の中で必ず再合流すること。
③繰返し構造：同じ処理をループする構造で、繰り返すかどうかの条件判断のタイミングで、前判定繰返しと、後判定繰返し（右図の2つの繰返し構造はどちらも後判定）がある。
例：モンスターの数だけ戦う＝前判定繰返し（0匹なら戦わない）
例：正しい値が得られるまで繰り返す＝後判定繰返し（得てから判断する）
※ 流れ図で繰返しを表現する場合、終了条件を書く場合と、継続条件を書く場合があるの注意
p.165 7-5-2 基本的なアルゴリズム
・【補足】変数：バリアブルの和訳。データを格納する領域に名前をつけたもの。データの入る箱のイメージ。データの型によって大きさが変わる（例：int 型変数の多くは 32 ビット）。
・【補足】代入：変数にデータを格納すること。すでにデータが代入されている変数にさらにデータを代入すると上書きされる。流れ図や疑似言語では「〇→変数」または「変数←〇」と表す。
・【補足】合計のアルゴリズムを箇条書き風に表すと：
①合計用の変数に0を代入
②データを1件得る
③得られていたら、合計用の変数に足し込み②から繰り返す（得られていなかったら繰り返し終了）
④ 合計用の変数のデータを表示
・【補足】探索（データが格納された配列などから特定のデータがあるか探すこと）のアルゴリズム（今回はあれば添字（インデクス）を表示）
①添字用の変数 i に1を代入
②探索用の変数 x にデータを代入
④ 配列の i 番目がxと等しければ「i 番目にある」と表示して終了
⑤ 等しくなければ、次のデータを見るために i に 1 加算し、データ数を超えなければ④から繰り返す
⑥ データ数を超えてしまったら「なし」と表示して終了

・【補足】整列：ソートともいい、複数のデータを値の順番に並べなおすこと。小から大の順にすることを昇順、大から小の順にすることを降順という。
例：通常の成績表は降順。ゴルフや100m競争の成績表は昇順
・【補足】整列アルゴリズム：整列は非常に多くの場面で行われるが、扱うデータや実行環境により複数のアルゴリズムの使い分けが行われている
・【補足】選択法（選択ソート）：低速だがコンピュータの負荷が小さいアルゴリズムで、ある程度、整列されているデータに用いると高速。
・【補足】交換法（バブルソート）：低速だがコンピュータの負荷が小さいアルゴリズムで、特に負荷の高い環境や、メモリの空きが乏しい環境でも用いることができる。
・【補足】高速ソート：上記以外にコンピュータの負荷が高いが高速なアルゴリズムもあり、ヒープソート、クイックソートなどが利用されている。
・【補足】計算量：アルゴリズムの実行に必要な計算の数と、データ量の関係のことで、合計や探索は件数に比例する。選択ソートやバブルソートは件数の2乗に比例する。ちなみに、配列から特定の要素を得る処理の計算量は1固定。
p.169 7-4-4 疑似言語で表すアルゴリズム
・【補足】疑似言語の記述形式：p.169 の表は試験画面で同じものを見られるので暗記不要。
・【補足】変数：疑似言語では事前に使用する変数を宣言する必要がある（宣言によって変数が用意＝確保される）。宣言の書式は「型名：変数名」。
例：「文字列型：学生名」「整数型：年齢」
・【補足】注釈：コメントともいい、実行に影響しないメモ書き。説明用で「／＊」と「＊／」で挟んで記述する。また「／／」から文末までという書式もある。
例：「整数型：年齢 /＊ 負の数にはならない ＊/」

今日の1問 スタックからの取り出しは？ ア ENQ イ DEQ ウ PUSH エ POP
次回予告： p.169「疑似言語の記述形式」の続きから