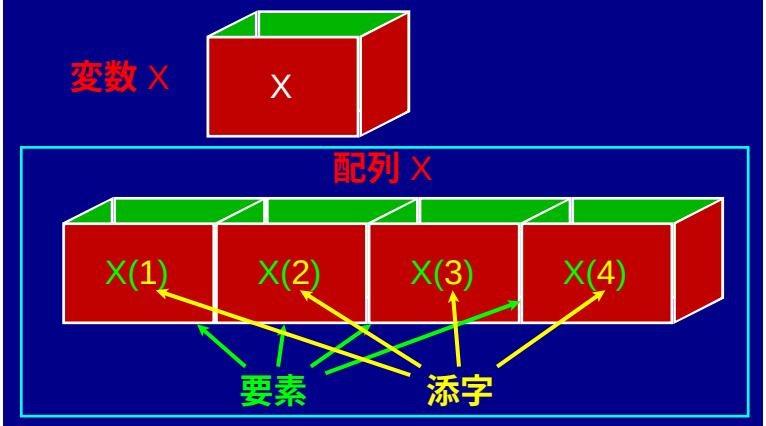
基本的なデータ構造

配列型
レコード型(構造体)
ポインタ
スタック
キュー(待ち行列)
リスト
ツリー構造(木構造)

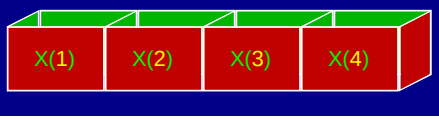
配列型



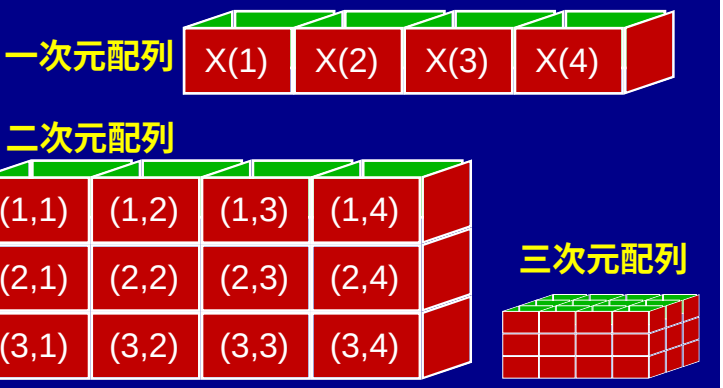
配列型

同じ型のデータを一括して扱う

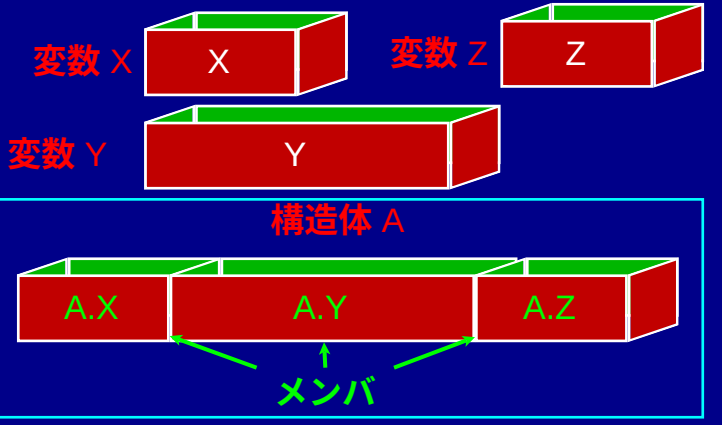
配列の要素
個々のデータ=変数と同様に使用可
添字
何番目の要素かを示す(整数)



配列型



レコード型(構造体)

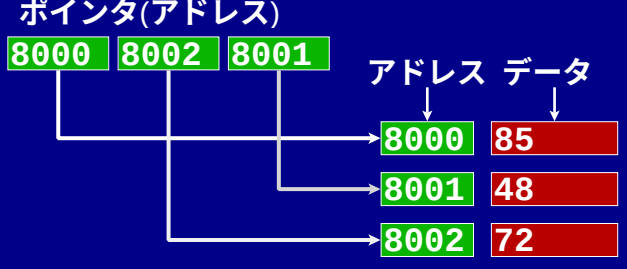


レコード型(構造体)

型の異なるデータを一括して扱う
大きさの異なるデータでもよい
(実数、整数、文字、文字列、配列)
メンバ、メンバ名
データの一つ一つ(メンバ)にそれを参照するために「名前」を付ける必要がある

ポインタ

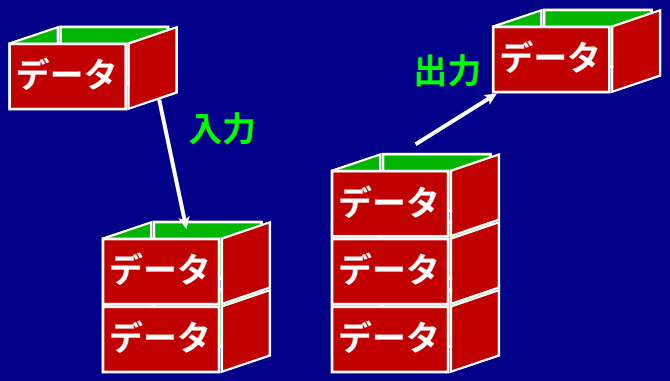
データが格納されている主記憶装置上のアドレス情報



ポインタ

高水準言語
論理的に連続するデータ(配列など)のアドレス情報(配列の添字など)
低水準言語
データが格納されている主記憶装置(メモリ)上の物理アドレス
ポインタ型変数(C言語)
アドレスを格納でき、論理的に扱うことができる変数

スタック

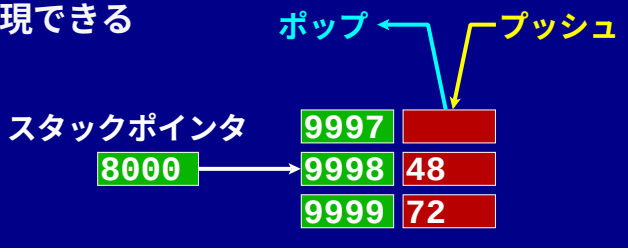


スタック

「洞窟」「輪投げの輪」

後から入力したものを先に出力(処理)する:
後入れ先出し(Last In First Out: LIFO)

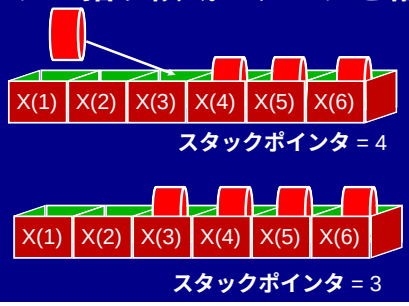
一次元配列と一つのスタックポインタで表現できる



スタックの管理

データの格納(push)

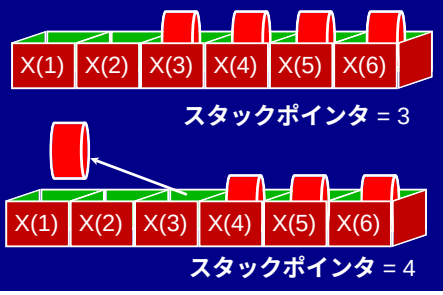
スタックポインタを「1」減らし、スタックポインタの指す領域にデータを格納する



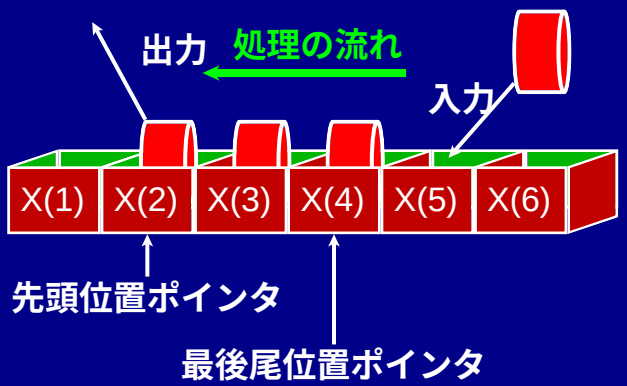
スタックの管理

データの取り出し(pop)

スタックポインタの指す領域からデータを取り出し、スタックポインタを「1」増やす



キュー(待ち行列)



キュー(待ち行列)

「トンネル」「窓口に並ぶ行列」

先に入力したものから順に出力(処理)する :
先入れ先出し(First In First Out: FIFO)

一次元配列で表現できる

データの「先頭位置を指すポインタ」と「最後尾を指すポインタ」を用意する

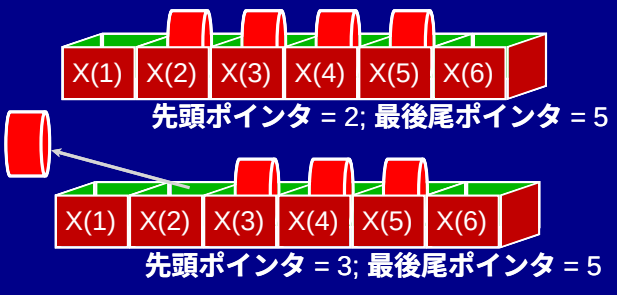
記憶領域が(論理的に)環状になっていると仮定して使用する

キュー(待ち行列)の管理

データを格納する

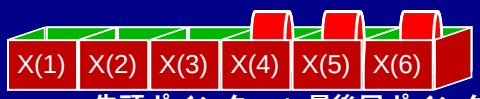
キュー(待ち行列)の管理

データを取り出す

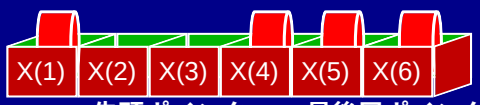


キュー(待ち行列)の管理

データの再配列を行うか、データ領域を環状にしてもちいる



先頭ポインタ = 4; 最後尾ポインタ = 6

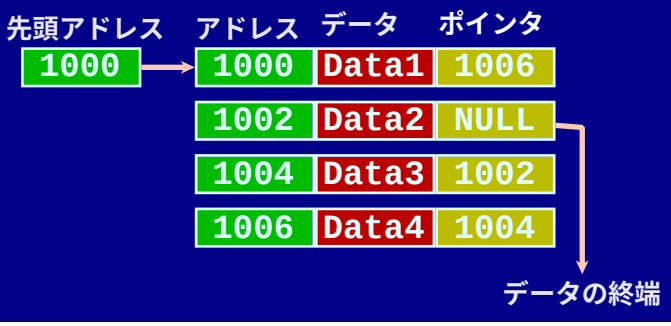


先頭ポインタ = 4; 最後尾ポインタ = 1

リスト

一群のデータをポインタでつなぐ

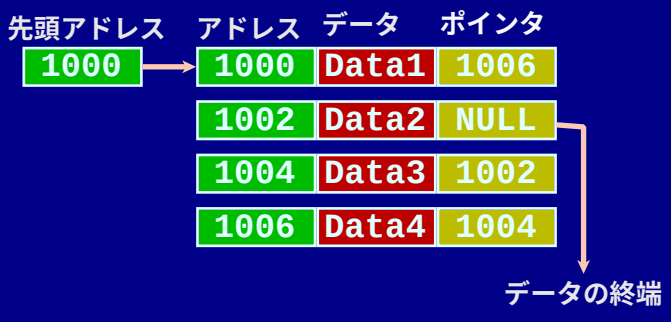
データの挿入・削除が容易



リスト

単リスト

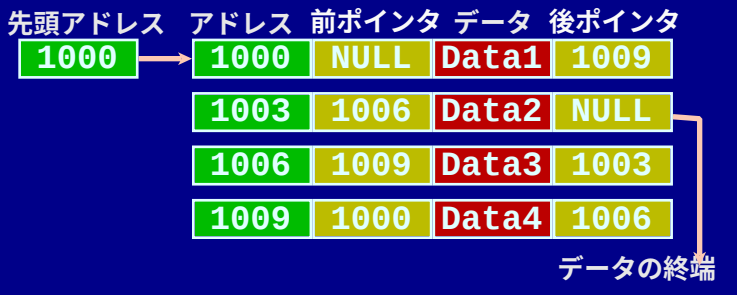
データ部とポインタ部から構成される



リスト

双リスト

前ポインタとデータ部、後ポインタ部から構成される



リスト

環状リスト

前ポインタ、後ポインタに先頭、終端を表すNULLポインタがなく環状に構成される

リストの管理

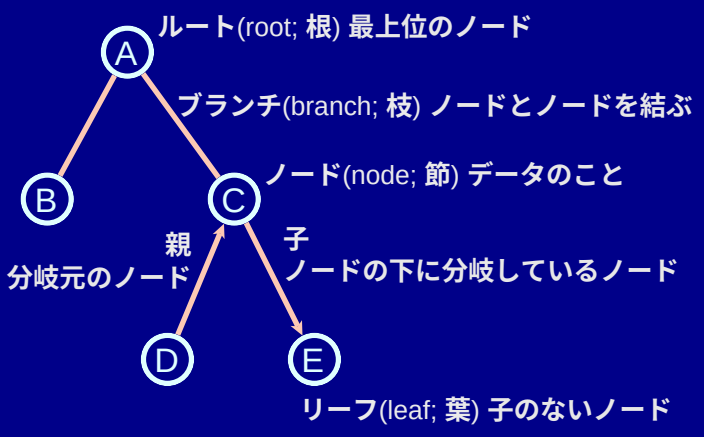
データの追加(挿入)

リストの管理

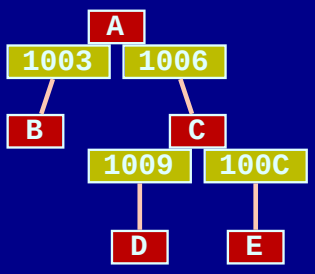
データの削除

定期的なデータの再配列が必要になる

ツリー構造(木構造)



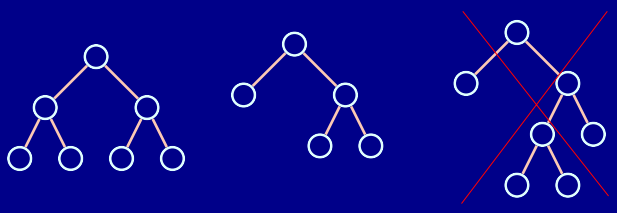
2分木



1000	1003	A	1006
1003	NULL	B	NULL
1006	1009	C	100C
1009	NULL	D	NULL
100C	NULL	E	NULL

完全2分木

ルートから全てのリーフに至るブランチの数がせいぜい1しか違わない2分木



2分木の走査

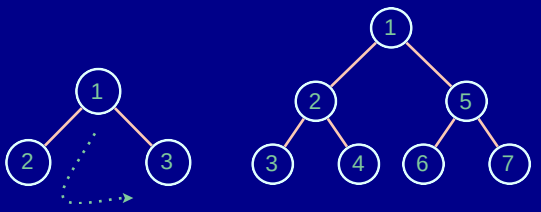
前順(行きがけ順)
親節→左部分木→右部分木

間順(通りがけ順)
左部分木→親節→右部分木

後順(帰りがけ順)
左部分木→右部分木→親節

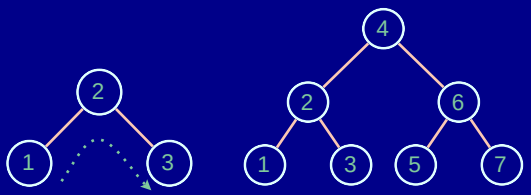
2分木の走査

前順(行きがけ順)
親節→左部分木→右部分木



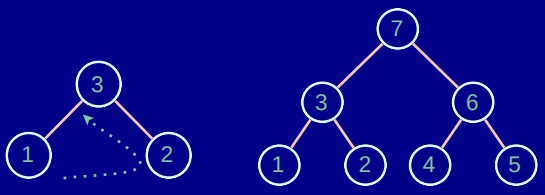
2分木の走査

間順(通りがけ順)
左部分木→親節→右部分木



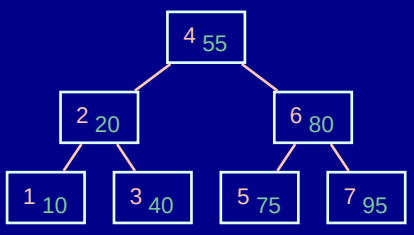
2分木の走査

後順(帰りがけ順)
左部分木→右部分木→親節



2分探索木

昇順データを間順に2分木に配置



各ノードの左は親より小さく右は親より大きい

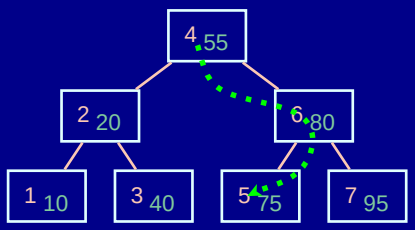
2分探索木

データの探索

1) ルートからスタート
2) 探索データとノード値を比較
3) 等しければ探索終了
4) 探索データ<ノード値→左ノードに移動
5) 探索データ>ノード値→右のーどに移動
6) リーフでなければ2)にもどる
7) リーフであれば探索データなし

2分探索木

データ{75}を探索



2分木の応用例(数式処理)

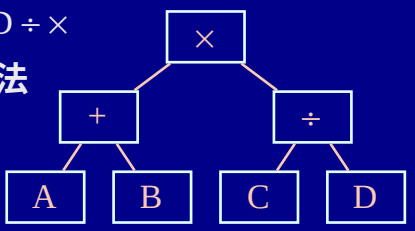
(A + B) × (C ÷ D)

前順: × + A B ÷ C D

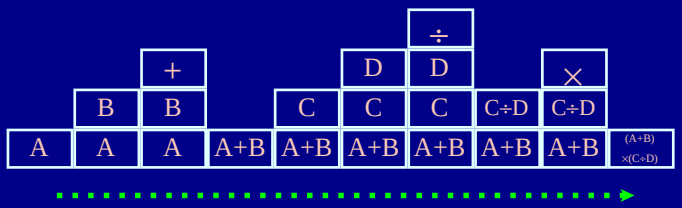
間順: A + B × C ÷ D

後順: A B + C D ÷ ×

逆ポーランド記法



逆ポーランド記法とスタック



配列の宣言

配列を使用するためには変数宣言と同様に配列の宣言を行う

- 配列の名前
- 配列の要素の型
- 配列の添字の範囲

配列の宣言と参照(Pascal)

宣言文

var 配列名: array [添字範囲] of 要素の型

添字範囲　[最小添字..最大添字]

二次元配列　[最小..最大, 最小..最大]

要素の参照

配列名[添字(整数値)]

配列名[添字(整数値), 添え字(整数値)]

配列の宣言と参照(Pascal)

例

```
var a: array [1..100] of integer;
  a[1], a[2], a[3], …, a[100]

var x: array [0..31, 0..31] of real;
  x[0,0], x[0,1], x[0, 2], …, x[0,31],
  x[1,0], x[1,1], x[1, 2], …, x[1,31],
   …,
  x[31,0], x[31,1], x[31,2], …, x[31,31]
```

配列の宣言と参照(FORTRAN)

宣言文

要素の型 配列名

DIMENSION 配列名(添字範囲)

添字範囲　(最小添字 : 最大添字)

二次元配列　(最小 : 最大, 最小 : 最大)

要素の参照

配列名(添字(整数値))

配列名(添字(整数値), 添え字(整数値))

配列の宣言と参照(FORTRAN)

例

```
INTERGER A
DIMENSION A(100)
  a[1], a[2], a[3], …, a[100]
DIMENSION X (0:31, 0:31)
  X(0,0), X(0,1), X(0, 2), …,X(0,31),
  X(1,0), X(1,1), X(1, 2), …, X(1,31),
   …,
  X(31,0), X(31,1),X(31,2), …,X(31,31)
```

配列の宣言と参照(c言語)

宣言文

- 要素の型 配列名[要素の数]
- 添字範囲 [0] から[要素の数–1]
- 二次元配列 [要素の数][要素の数]

要素の参照

- 配列名[添字(整数値)]
- 配列名[添字(整数値)][添え字(整数値)]

配列の宣言と参照(c言語)

例

int a[100];
a[0], a[1], a[2], …, a[99]

double x[32][32];
x[0][0], x[0][1], x[0][2], …, x[0][31],
x[1][0], x[1][1], x[1][2], …, x[1][31],
…,
x[31][0], x[31][1], x[31][2], …, x[31][31]

[例題]

点数を読み込んでその平均値を出力するプログラム
ただし、データの個数は既にわかっているものとする

例題(Pascal)

```
program average(input, output);
const n = 10;
var x: array [1..n] of integer;
    i, sum: interger;
    a: real;
begin
  for i:= 1 to n do read(x[i]);
  sum:=0;
  for i:= 1 to n do sum:=sum+x[i];
  a:= sum/n;
  writeln(a)
end.
```

例題(c言語)

```
#include <stdio.h>
#define N 10
int main(void){
  int x[N];
  int i, sum;
  double a;
  for(i=0;i<N;i++)scanf("%d",&x[i]);
  sum=0;
  for(i=0;i<N;i++) sum+=x[i];
  a=(double) sum/ (double) N;
  printf("%lf\n", a);
  return 0;
}
```

例題(FORTRAN)

```
      PROGRAM AVERAG
      PARAMETER(N=10)
      INTEGER I, X, SUM
      DOUBLE PRECISION A
      DIMENSION X(1:N)
      READ(5, *) (X(I), I=1, N)
      SUM=0
      DO 10 I=1, N
   10 SUM=SUM+X(I)
      A=DFLOAT(SUM)/DFLOAT(N)
      WRITE(6,*) A
      STOP
      END
```