Laborator 3

Stive și paletizare

As. ing. Alexandru Dumitrache
As. ing. Raluca Tudorie
www.scr.cimr.pub.ro

Operația *Pick and Place* din laboratorul precedent poate fi aplicată pentru a aranja mai multe piese identice pe o stivă, pe o paletă, într-un depozit etc.

O abordare posibilă, dar ineficientă, ar fi învățarea fiecărui punct robot. Este posibilă învățarea unui singur punct (originea), și cunoscând dimensiunile și orientarea depozitului, celelalte puncte pot fi calculate.

Pentru aceasta se poate folosi funcția V^+ `SHIFT`, care modifică o locație în spațiul cartezian *World*, doar în translație (orientarea se păstrează).

```
SET loc.new = SHIFT(loc.old BY dx, dy, dz)
```

Exemplu: Se dorește preluarea unor piese identice dintr-o stivă verticală pentru care s-a învățat un singur punct numit `baza` stivei. Se cunoaște grosimea pieselor `h.piesa` și numărul de piese din stivă `nr.piese`.

Poziția de prindere pentru piesa din vârful stivei este:

```
SET pick = SHIFT (baza BY 0, 0, (nr.piese-1) * h.piesa)
```

Stive verticale

Se dă o stivă cu `n` piese de înălțime `h`, pentru care se cunoaște punctul robot de la baza acesteia, `st.a`. Se dorește mutarea tuturor pieselor în stiva a doua, pentru care se cunoaște punctul robot `st.b` (Fig. 3.1).

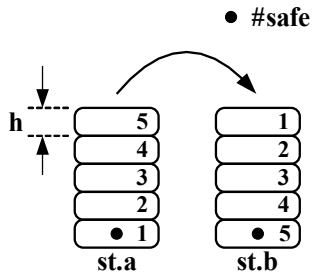

Figura 3.1: *Stive verticale*

Aplicația începe și se încheie cu robotul în poziția `#safe`, cu gripperul deschis.

Observație: Dacă punctul de prindere a piesei nu a fost învățat foarte precis, se recomandă așezarea piesei puțin mai sus decât poziția învățată (sau calculată), marginea de siguranță putând fi de 0.5 mm.

Vom rescrie programul `pick.place` sub forma unei subrutine:

```
.PROGRAM pick.place(pick, place)
    ; Laboratorul 3 - Subrutina pick.place

    AUTO z.pick, z.place
    z.pick = 80
    z.place = 80

    PARAMETER HAND.TIME = 0.2
    OPEN
    APPRO pick, z.pick
    BREAK
    SPEED 50
    MOVES pick
    CLOSEI
    SPEED 30
    DEPARTS z.pick
    BREAK
```

```
    APPRO place, z.place
    BREAK
    SPEED 20
    APPROS place, 0.5  ; marginea de siguranta
    OPENI
    SPEED 50
    DEPARTS z.place
    BREAK
.END


.PROGRAM stiva.vert()
    ; Laboratorul 3 - Stiva verticala

    GLOBAL #safe, st.a, st.b   ; locatii robot
    AUTO n, h, i, r
    AUTO pick, place

    n = 5;
    h = 4.25;

    SPEED 100 ALWAYS
    MOVET #safe, TRUE   ; miscarea incepe din #safe
    BREAK               ; cu gripper-ul deschis

    FOR i = 1 TO n
         r = n - i + 1  ; r = nr. pieselor din stiva st.a
         SET pick = SHIFT(st.a BY 0, 0, (r - 1) * h);
         SET place = SHIFT(st.b BY 0, 0, (i - 1) * h);

         CALL pick.place(pick, place)
    END

    MOVE #safe   ; intoarcere in #safe
.END
```

Stive tridimensionale

Se dă o stivă verticală cu M piese ($M = L \times C \times N$), numerotate de la 1 la M. Dorim să așezăm piesele într-o stivă tridimensională (paletă) cu L linii, C coloane și N nivele (Fig. 3.2 și 3.3).

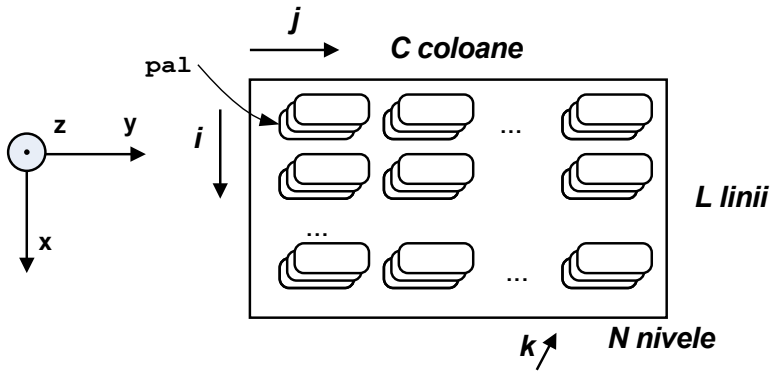

Figura 3.2: *Stiva tridimensională*

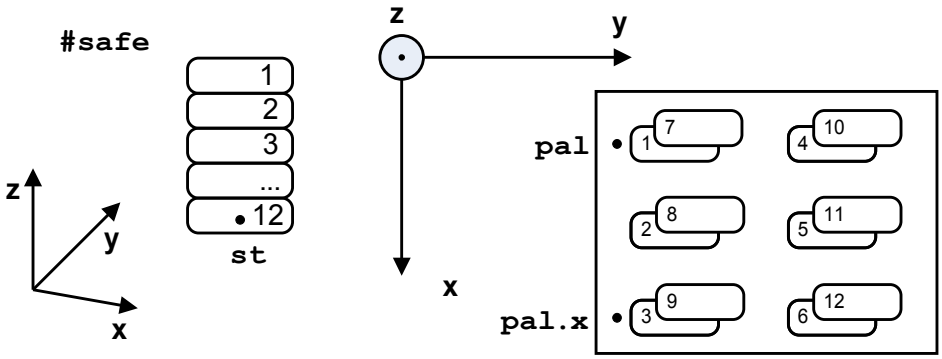

Figura 3.3: *Particularizare pentru $L = 3, C = 2, N = 2$, completare NCL*

Se cunosc:

- baza stivei verticale `st`;
- prima poziție din stiva 3D: `pal`, poziția $(0, 0, 0)$;
- înălțimea unei piese: `dz`;
- distanțele între 2 linii și 2 coloane consecutive pe paletă: `dx` și `dy`;
- indicii în matrice (i, j, k) corespund axelor carteziene World (X, Y, Z).

Observație: În practică, axele paletei nu sunt aliniate cu axele robotului.

Piesele pot fi așezate în stiva tridimesională în mai multe moduri. Dorim să exprimăm poziția pe care va fi așezată pe paletă piesa cu numărul p în matrice, prin indicii (i, j și k).

Definiții:

- Linia x = totalitatea elementelor din matrice care au coordonata $i = x$;
- Coloana x = totalitatea elementelor care au coordonata $j = x$;
- Nivelul x = totalitatea elementelor care au coordonata $k = x$.

Formulele de calcul pentru diverse moduri de completare a stivelor sunt:

- Mod de completare NCL (nivel, coloană, linie):

$$\begin{aligned} i_0^{NCL} &= (p-1) \bmod L \\ j_0^{NCL} &= \left\lfloor \frac{p-1}{L} \right\rfloor \bmod C \\ k_0^{NCL} &= \left\lfloor \frac{p-1}{L \cdot C} \right\rfloor \end{aligned}$$

- Mod de completare NLC (nivel, linie, coloană):

$$\begin{aligned} i_0^{NLC} &= \left\lfloor \frac{p-1}{C} \right\rfloor \bmod L \\ j_0^{NLC} &= (p-1) \bmod C \\ k_0^{NLC} &= \left\lfloor \frac{p-1}{L \cdot C} \right\rfloor \end{aligned}$$

- Mod de completare CLN (coloană, linie, nivel):

$$\begin{aligned} i_0^{CLN} &= \left\lfloor \frac{p-1}{N} \right\rfloor \bmod L \\ j_0^{CLN} &= \left\lfloor \frac{p-1}{L \cdot N} \right\rfloor \\ k_0^{CLN} &= (p-1) \bmod N \end{aligned}$$

În ecuațiile de mai sus, numărul piesei este *1-based*, însă poziția în matrice este *0-based*. Indicii *0-based* au avantajul de a simplifica formulele, însă este de preferat ca numărul piesei să fie *1-based*. Astfel, utilizatorul programului

va putea specifica numărul de piese din stivă sau depozit, și nu un contor a cărui semnificație poate fi neintuitivă.

În V^+ , $\lfloor x \rfloor$ se traduce prin `INT(x)` (valabil doar pentru $x \geq 0$), iar a mod b este `a MOD b`.

După ce se cunosc distanțele dx, dy și dz și poziția piesei p în matrice (coordonatele zero-based i, j, k), poziția de prindere este:

```
SET pick = SHIFT(pal BY i*dx, j*dy, k*dz)
```

Cazurile în care indicii i, j și k nu corespund cu axele de coordonate X, Y și Z în această ordine, sau au orientări diferite, sunt propuse ca temă.

Corespondențe: $M =$ `nr.piese`, $L =$ `nl`, $C =$ `nc`, $N =$ `nn`.

```
.PROGRAM palet.ideal()
    ; Laboratorul 3 - Paletizare (cazul ideal)
    ; Nu se ruleaza pe robot!

    GLOBAL #safe, st, pal ;locatii robot
    AUTO dx,dy,dz
    AUTO i,j,k,p,r,nr.piese
    AUTO nl,nc,nn        ; nr.linii, nr.coloane, nr.nivele
    AUTO pick, place

    nl = 3
    nc = 2
    nn = 2
    nr.piese = nl*nc*nn

    dx = 50
    dy = 50
    dz = 10

    SPEED 100 ALWAYS
    OPEN
    MOVE #safe
    BREAK
```

```
    FOR p = 1 TO nr.piese     ; p = indicele piesei curente
        r = nr.piese - p + 1  ; r = nr. pieselor din stiva

        i = (p-1) MOD nl
        j = INT((p-1)/nl) MOD nc
        k = INT((p-1)/(nl*nc))
        TYPE i, ", ", j, ", ", k

        SET pick = SHIFT(st BY 0,0,(r-1)*dz)
        SET place = SHIFT(pal BY i*dx, j*dy, k*dz)

        CALL pick.place(pick, place)
    END

    MOVE #safe        ; intoarcere in #safe
.END
```

Stive 3D rotite în jurul axei Z

În majoritatea cazurilor practice, stivele nu sunt perfect paralele cu axele robotului. La roboții Cobra, planul XY al stivei este întotdeauna paralel cu planul XY al robotului, însă stiva este rotită pe RZ cu un unghi α (cunoscut sau nu). Roboții Viper pot lucra și pe plan înclinat.

Pentru a aborda această situație avem nevoie de câteva noțiuni despre compunerea transformărilor.

O transformare este dată de 6 parametri:

```
SET h = TRANS(x, y, z, yaw, pitch, roll)
```

Primii 3 reprezintă poziția, iar ultimii 3 reprezintă orientarea în spațiu. În controller-ul robot, ele sunt reprezentate prin matrici omogene (HTM - Homogeneous Transformation Matrix), de dimensiune 4×4, cu structura:

$$H = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.1}$$

Submatricea 3x3 din colțul stânga sus este componenta de rotație, dată de unghiurile Euler *yaw*, *pitch* și *roll* - convenția ZYZ'. Parametrul *yaw* reprezintă rotația în jurul axei OZ, *pitch* este rotația în jurul axei OY după ce *yaw* a fost aplicat, iar *roll* reprezintă rotația în jurul noii axe OZ obținută după aplicarea rotațiilor *yaw* și *pitch*.

Formula de calcul a matricii de rotație descrise de *yaw*, *pitch* și *roll* este:

$$R_{ypr} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \mathcal{R}_Z(yaw) \cdot \mathcal{R}_Y(pitch) \cdot \mathcal{R}_Z(roll) \tag{3.2}$$

V^+ folosește această reprezentare internă pentru transformări. Pentru a economisi resurse, ultima linie nu mai este memorată, deoarece este întotdeauna [0 0 0 1].

Compunerea transformărilor se realizează cu operatorul $\gg$: $\ll$. V^+ calculează rezultatul prin înmulțirea matricilor HTM:

```
SET trans.c = trans.a:trans.b
```

Observație: Nu se pun spații nici înainte, nici după operatorul $\gg$: $\ll$!

Transformări elementare:

- `TRANS(dx,dy,dz)`: translație
- `RX(ang)`: rotație în jurul axei X, unghiul este exprimat în grade
- `RY(ang)`: rotație în jurul axei Y
- `RZ(ang)`: rotație în jurul axei Z
- `NULL`: transformarea nulă (matricea unitate I_4)

Formule elementare de compunere:

- `loc:TRANS(dx,dy,dz)`: translație pe sistemul de coordonate Tool
- `loc:RZ(45)`: rotația efectorului terminal în jurul axei Z_{tool}
- `loc:TRANS(0,0,10):RZ(45)`: înșurubare
- `SHIFT(loc BY dx,dy,dz)` echivalent cu `TRANS(dx,dy,dz):loc`

Alte funcții utile pentru lucrul cu transformări:

- `DISTANCE(a,b)`: returnează distanța între cele două locații (nu ține cont de orientare)
- `DX(loc)`, `DY(loc)`, `DZ(loc)`: extragerea componentelor X, Y, Z
- `DECOMPOSE elem[] = loc`: extragerea celor 6 componente din transformarea `loc` în vectorul `elem[]`, de la `elem[0]` la `elem[5]`

Fie sistemul de coordonate World al robotului (X_0, Y_0, Z_0) și fie un sistem de referință local (x, y, z), rotit față de World cu un unghi `ang` în jurul azei Z_0 (Fig. 3.4). Se cunosc: punctul robot `loc.old`, unghiul `ang` și offset-urile `dx` și `dy`; se dorește determinarea punctului `loc.new` .

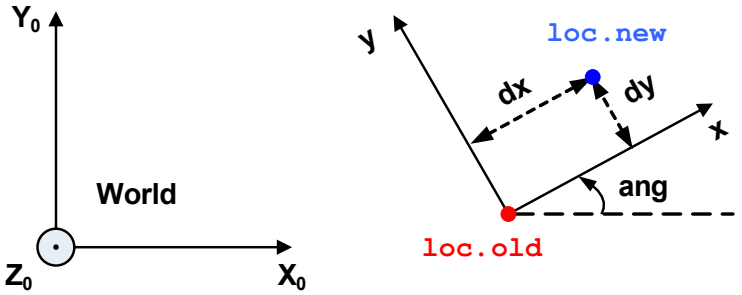

Figura 3.4: *Sistem de coordonate rotit în jurul axei Z*

Folosind formula rotației în plan, se poate scrie:

```
SET loc.new = SHIFT(loc.old BY dx*COS(ang) - dy*SIN(ang),
                               dx*SIN(ang) + dy*COS(ang), 0)
```

Folosind compunerea transformărilor, putem obține o soluție mai elegantă.

Știind că `SHIFT(loc BY dx,dy,dz)` este echivalent cu `TRANS(dx,dy,dz):loc`, rezultă formula pentru `loc.new`:

```
SET loc.new = RZ(ang):TRANS(dx,dy,0):RZ(-ang):loc.old
```

Astfel, dacă se cunoaște unghiul *ang* dintre paletă și robot, locația în care va fi așezat elementul (i, j, k) se poate calcula cu formula:

```
SET place = RZ(ang):TRANS(i*dx,j*dy,k*dz):RZ(-ang):pal
```

Mai rămâne problema determinării unghiului *ang*. Acest lucru se poate face ușor învățând un al doilea punct pe paletă, care să indice direcția X, de exemplu pe ultima linie, în poziția $(L-1,\ 0,\ 0)$. Vom denumi acest punct `pal.x` (Fig. 3.3) și vom calcula unghiul cu:

```
ang = ATAN2(DY(pal.x)-DY(pal), DX(pal.x)-DX(pal))
```

Punctul `pal.x` poate fi folosit de asemenea pentru calculul offset-ului `dx`. Putem presupune că `dy = dx`. Mai întâi ne vom asigura că `pal.x` și `pal` sunt la același nivel (au aceeași coordonată Z):

```
SET pal.x.ajustat = SHIFT(pal.x BY 0, 0, DZ(pal)-DZ(pal.x))
```

Rezultă offset-ul `dx`:

```
dx = DISTANCE(pal, pal.x.ajustat) / (nl - 1)
```

Acum putem modifica programul de paletizare a.î să poată fi rulat pe robot.

Se cunosc:

- M piese, $M = L \times C \times N$
- `st` - baza stivei verticale;
- `pal` - poziția $(0, 0, 0)$ pe paletă;
- `pal.x` - poziția $(L - 1,\ 0,\ 0)$, indică direcția axei X;
- `dz` - înălțimea unei piese;
- se presupune `dx = dy`;
- indicii în matrice (i, j, k) corespund axelor carteziene de pe paletă (x, y, z). Sistemul de coordonate este rotit față de World, în jurul lui Z, cu un unghi `ang` care va fi calculat.

```
.PROGRAM paletizare()
    ; Laboratorul 3 - Paletizare
    ; Poate fi rulat pe robot

    GLOBAL #safe, st, pal, pal.x ; locatii robot
    AUTO dx, dy, dz, ang
    AUTO i, j, k, p, r, nr.piese
    AUTO nl, nc, nn    ; nr.linii, nr.coloane, nr.nivele
    AUTO pick, place
    AUTO pal.x.ajustat

    nl = 3
    nc = 2
    nn = 2
    nr.piese = nl*nc*nn
```

```
    ; Aducem pal.x la acelasi nivel cu pal
    SET pal.x.ajustat = SHIFT(pal.x BY 0,0,DZ(pal)-DZ(pal.x))

    ; Calculam distanta intre 2 piese
    dx = DISTANCE(pal, pal.x.ajustat) / (nl - 1)
    dy = dx
    dz = 4.3
    TYPE "dx = ", dx

    ; Calculam unghiul dintre paleta si robot (RZ)
    ang = ATAN2(DY(pal.x)-DY(pal), DX(pal.x)-DX(pal))
    TYPE "ang = ", ang

    SPEED 100 ALWAYS
    OPEN
    MOVE #safe
    BREAK

    FOR p = 1 TO nr.piese ; p = indicele piesei curente
        r = nr.piese-p+1  ; r = nr. pieselor din stiva
        i = (p-1) MOD nl
        j = INT((p-1)/nl) MOD nc
        k = INT((p-1)/(nl*nc))
        TYPE i, ", ", j, ", ", k

        SET pick = SHIFT(st BY 0,0,(r-1)*dz)
        SET place = RZ(ang):TRANS(i*dx,j*dy,k*dz):RZ(-ang):pal

        CALL pick.place(pick, place)
    END

    MOVE #safe ; intoarcere in #safe
.END
```

Probleme rezolvate

Căsuța

Se consideră un robot industrial articulat vertical care are la dispoziție două palete; de pe prima paletă va prelua piesele și le va așeza pe a doua paletă.

Pe prima paletă există două stive verticale, fiecare având 6 piese sub forma literei I. Prima stivă conține piese roșii, iar cea de-a doua conține piese galbene. Se cunosc punctele de prindere ale pieselor aflate în vârful stivelor (`a` pentru stiva roșie respectiv `b` pentru stiva galbenă).

Plasarea pieselor se va face alternând două piese roșii și două piese galbene dispuse sub forma unui pătrat, în jurul unui punct robot numit c (Fig. 3.6). Prima dată se pun două piese roșii paralele, apoi se pun două piese galbene, perpendicular pe cele roșii, iar ciclul se repetă pentru toate cele 12 piese.

Aplicația începe și se încheie cu robotul în poziția `#safe`, cu gripperul deschis. Brațul robot va fi folosit în configurația `LEFTY`, `ABOVE` și `NOFLIP`.

Distanța între două găuri este de `d` = 1.25 inch; 1 inch = 25.4 mm. Înălțimea unei piese este `h` = 4.25 mm.

Se consideră că punctul de prindere este învățat exact în centrul piesei, iar în punctul `c` piesa este paralelă cu axa X a robotului.

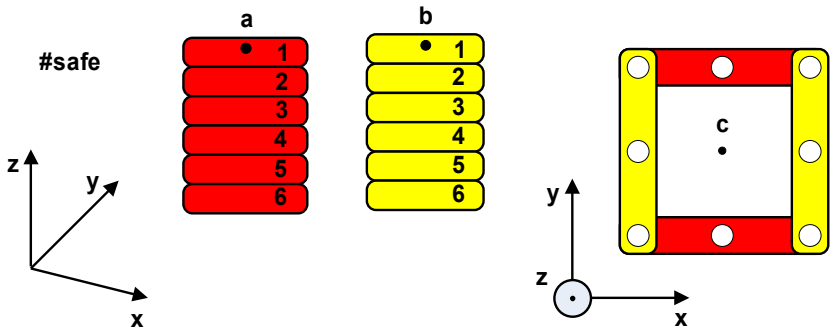

Figura 3.5: *Căsuța*

Soluție: Problema va fi rezolvată în trei pași principali: `FOR i = 0 TO 2`. La fiecare pas vor fi așezate 4 piese, ale căror puncte de prindere și de plasare vor fi memorate în vectorii de tip `AUTO`: `sursa[4]` și `desti[4]`. Piesele din stiva `a` vor fi translatate pe Y cu $\pm d$, iar piesele din stiva `b` vor fi translatate pe X cu $\pm d$ și rotite cu 90° în jurul axei Z_{tool}.

```
.PROGRAM casuta()
    GLOBAL a, b, c, #safe
    AUTO sursa[4]
    AUTO desti[4]
    AUTO i, j, d, h
    d = 1.25 * 25.4
    h = 4.25

    SPEED 100 ALWAYS
    MOVET #safe, TRUE
    BREAK

    LEFTY
    ABOVE
    NOFLIP

    FOR i = 0 TO 2
        FOR j = 0 TO 1
            SET sursa[j] = SHIFT(a by 0, 0, - h * (2*i + j))
            SET desti[j] = SHIFT(c by 0, d * SIGN(j-0.5), h*(i*2))
        END

        FOR j = 0 TO 1
            SET sursa[j+2] = SHIFT (b by 0, 0, - h * (2*i + j))
            SET desti[j+2] = SHIFT (c by d*SIGN(j-0.5), 0, h*(i*2+1))
                                                              :RZ(90)
        END

        FOR j = 0 TO 3
            CALL pick.place(sursa[j], desti[j])
        END
    END

    MOVET #safe, TRUE
.END
```

Spirala

Se consideră un robot industrial articulat vertical care are la dispoziție două palete; de pe prima palată va prelua piesele și le va depune pe a doua paletă.

Pe prima paletă există două stive verticale fiecare având `n/2` piese sub forma literei I (`n` este număr par). O stiva conține piese roșii iar cealaltă conține piese galbene. Se cunosc punctele de prindere ale pieselor aflate în vârful stivelor (`a` pentru stiva roșie și `b` pentru stiva galbenă). Se consideră că punctele de prindere sunt învățate exact în centrul pieselor.

Plasarea pieselor se va face alternând o piesă roșie și una galbenă, în jurul unui ax aflat în punctul `c`, sub forma unei spirale. Prima piesă va fi plasată în punctul `c`; a doua piesă va fi așezată deasupra ei și rotită cu un unghi α; a treia piesă va fi rotită cu 2α; ultima piesă va fi rotită cu unghiul $total.ang = 360\,°$.

Aplicația începe și se încheie cu robotul în poziția `#safe`, cu gripperul deschis. Brațul robot va fi folosit în configurația `LEFTY`, `ABOVE` și `FLIP`.

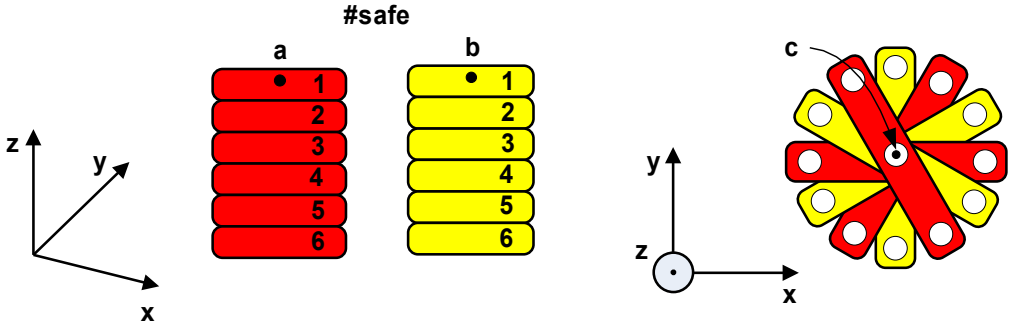

Figura 3.6: *Spirala*

Soluție: Vom numerota piesele de la 1 la `n`, în ordinea în care vor fi așezate. Pentru simplificarea adresării în cele două stive, vom folosi indicele zero-based al piesei care urmează să fie așezată (`i = p-1`) și indicele zero-based al piesei care urmează să fie luată din stivă (`is = INT(i/2)`). Piesele care au indicele `i` par vor fi luate din stiva `a`, iar celelalte, din stiva `b`.

Pentru a realiza rotația progresivă a pieselor, vom observa că `i / (n-1)` variază de la 0 la 1, deci vom înmulți această cantitate cu `total.ang`. Rotația se va realiza în jurul axei Z_{tool}.

Prin specificarea configurației `LEFTY / ABOVE / NOFLIP`, programul va putea rula atât pe Cobra (unde `ABOVE` și `NOFLIP` sunt ignorați), cât și pe robotul Viper din punctul `#safe` învățat în poziție verticală.

```
.PROGRAM spirala()
    GLOBAL a, b, c, #safe
    AUTO pick, place
    AUTO p, i, is, h, total.ang
    h = 4.25
    total.ang = 360
    n = 12

    SPEED 100 ALWAYS
    MOVET #safe, TRUE
    BREAK

    LEFTY
    ABOVE
    NOFLIP

    FOR p = 1 TO n
        i = p-1
        is = INT(i/2)
        IF i MOD 2 == 0 THEN
            SET pick = SHIFT (a BY 0,0,-h*is)
            SET place = SHIFT (c BY 0, 0, h*i)
                                :RZ(i * total.ang / (n-1))

        ELSE
            SET pick = SHIFT (b BY 0,0,-h*is)
            SET place = SHIFT (c BY 0,0,h*i)
                                :RZ(i * total.ang / (n-1))
        END

        CALL pick.place(pick, place)
    END

    MOVET #safe, TRUE
.END
```

Probleme propuse

Mutare inversă

Modificați programele `casuta` și `spirala`, astfel încât piesele să fie așezate în ordine inversă, în pozițiile inițiale.

Reordonare

Se consideră o stivă verticală pentru care este învățată poziția de la bază `p1`. În stivă se află `n` piese. Se dorește plasarea pieselor de-a lungul axei X și apoi reordonarea lor într-o stivă verticală, ca în Fig. 3.7 (invers față de ordinea de la început). Punctele învățate sunt `#safe` și `p1`. Se cunosc: distanța între 2 piese pe orizontală `dx` = 5 cm, înălțimea unei piese `h` = 5 mm.

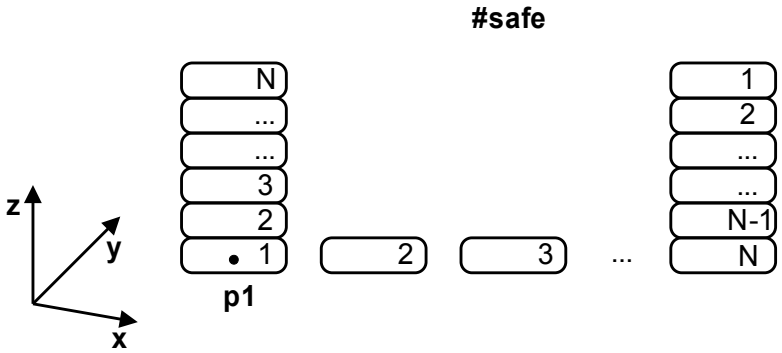

Figura 3.7: *Reordonare*

Alternanță

Se consideră două stive în care sunt plasate 10 piese roșii și 10 piese galbene. Se dorește prima dată plasarea tuturor pieselor roșii și apoi plasarea tuturor pieselor galbene pe o paletă 4 x 5, conform Fig. 3.8. Completarea se va face:

1. Prin completarea liniilor și apoi a coloanelor;
2. Prin completarea coloanelor și apoi a liniilor.

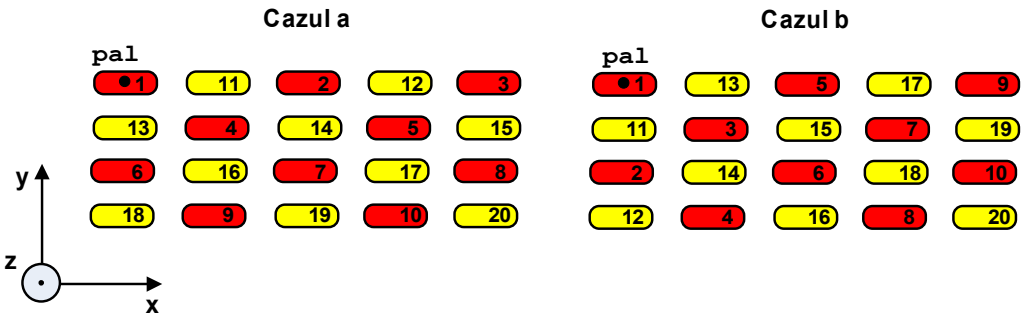

Figura 3.8: *Alternanță*

Distanțele `dx` și `dy` au valorile 4 cm respectiv 5 cm. Punctele învățate cu comanda monitor `.here` sunt `st.r` - la baza stivei roșii, `st.g` - la baza stivei galbene, `pal` - în colțul stânga sus al paletei și `#safe` - aflat în afara spațiului de lucru. Aplicația începe și se încheie în punctul `#safe`, cu gripperul deschis.

Diagonală

Se consideră un robot industrial articulat vertical care are la dispoziție o paletă bidimensională pe care sunt plasate 30 de piese (Fig. 3.9). Dorim plasarea primelor 15 piese într-o stivă verticală pentru care este cunoscută poziția de la bază `st.a` și apoi a următoarelor 15 într-o altă stivă unde cunoaștem poziția de la bază `st.b`. Piesele vor fi preluate pe diagonală începând cu poziția `pal`. Sunt 6 coloane și 5 linii.

Aplicația începe și se încheie cu robotul în poziția `#safe`, cu gripperul deschis, iar operația de prindere și așezare a pieselor se realizează în configurația `LEFTY` și `ABOVE`.

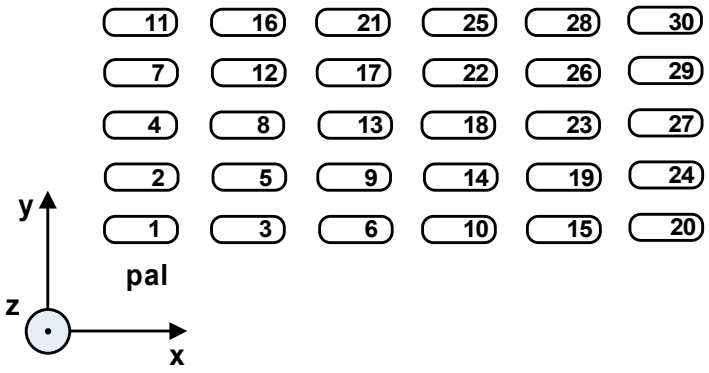

Figura 3.9: *Diagonală*

Problema turnurilor din Hanoi

Fie `n` discuri de diametre diferite, plasate unul peste altul în ordinea mărimii în prima stivă, conform Fig. 3.10.

Se dorește mutarea tuturor pieselor în stiva a doua, folosind ca stivă auxiliară pe cea de-a treia. Piesele cu diametru mai mic se vor pune întotdeauna deasupra celor cu diametru mai mare.

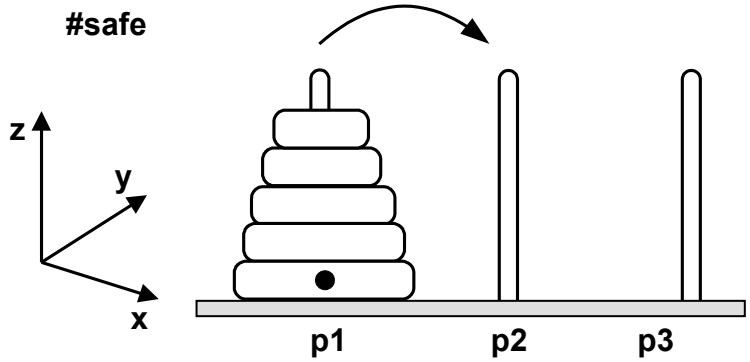

Figura 3.10: *Problema turnurilor din Hanoi*

Se cunosc locațiile `p1` și `p3`. Stiva `p2` se află exact la jumătate, între `p1` și `p3`. Aplicația începe și se încheie în punctul `#safe` cu gripperul deschis.