```
1
2   // Omówienie organizacji danych w pamięci
3
4       a:  1000: |     3 | : Integer (4B)
5       b:  1004: |     0 | : ^Integer (4B)
6       p:  1008: |  1000 | : Integer(4B)
7           1012: | | | | |
8           1016: | | | | |
9           1020: | | | | |
10
11
12  TYPE
13    PInteger: ^Integer;
14  VAR
15    a : Integer;
16    b : Integer;
17    p : ^Integer;
18
19  BEGIN
20    a := 3;
21    b := 0;
22    //  a = 3     zmienna
23    // @a = 1000  adres zmiennej
24
25    p := @a; // wskaźnik p wskazuje zmienną a         (pkt.4)
26
27    //  p = 1000
28    // @p = 1008
29    // p^ = 3       // teoretycznie (1000)^ = 3
30    // @(p^) = 1000 // zweryfikować
31
32    p^ := 5;                                    //  (pkt.5)
33
34      a:  1000: |     5 | : Integer (4B)
35      b:  1004: |     0 | : Integer (4B)
36      p:  1008: |  1000 | : ^Integer (4B)
37          1012: | | | | |
38
39    WriteLn('Zmienna a: ',a);                   //  (pkt.6)
40    WriteLn('Przez wskaznik p^: ',p^);
41
42    p := @b;                                    //  (pkt.7)
43    p^ := 7;
44
45      a:  1000: |     5 | : Integer (4B)
46      b:  1004: |     7 | : Integer (4B)
47      p:  1008: |  1004 | : ^Integer (4B)
48          1012: | | | | |
49
50    //  p = 1004
51    // @p = 1008
52    // p^ = 0
53
54      new(p); // przydzielenie pamięci           //  (pkt.8)
55
56    //  p = 5600
57    // @p = 1008
```

```
58     // p^ = ?
59
60       a:  1000: |     5 | : Integer (4B)
61       b:  1004: |     7 | : Integer (4B)
62       p:  1008: |  5600 | : ^Integer (4B)
63           1012: | | | | |
64           ...             // Pamięć przydzielana dynamicznie:
65           5600: |  ???  | : Integer
66
67       p^ = 123;                                  //  (pkt.9)
68       WriteLn(p^);
69       a := p^;
70
71       a:  1000: |   123 | : Integer (4B)
72       b:  1004: |     0 | : Integer (4B)
73       p:  1008: |  5600 | : ^Integer (4B)
74           1012: | | | | |
75           ...             // Pamięć przydzielana dynamicznie:
76           5600: |   123 | : Integer (4B)
77
78       dispose(p);         //  (pkt.10)
79       p := NIL;           //  (pkt.11)
80
81       a:  1000: |   123 | : Integer (4B)
82       b:  1004: |     0 | : Integer (4B)
83       p:  1008: |  NIL  | : ^Integer (4B)
84           1012: | | | | |
85           ...             // Pamięć przydzielana dynamicznie:
86           5600: | | | | |
87
88
89
90   TYPE
91     PInteger: ^Integer;
92   VAR
93     a,b : Integer;
94     p : ^Integer;
95     p1 : PInteger;
96     pp : ^PInteger;
97   BEGIN
98
99       a:  1000: |     3 |  // a:Integer (4B)
100      b:  1004: |  ???  |
101      p:  1008: |  1000 |
102      p1: 1012: |  ???  |
103      pp: 1016: |  ???  |
104          1020: | | | | |
105
106      pp := @p;
107
108      a:  1000: |     3 |  // a:Integer (4B)
109      b:  1004: |  ???  |
110      p:  1008: |  1000 |
111      p1: 1012: |  ???  |
112      pp: 1016: |  1008 |
113          1020: | | | | |
114
```

```
115     //  p  = 1000
116     // @p  = 1008
117     //  p^ = 3
118     //  pp = 1008
119     // @pp = 1016
120     // pp^ = 1000
121     // pp^^ = 3
122
123
124
125       //  (pkt.10)
126   PROCEDURE delete(var p:^Integer);
127   BEGIN
128       dispose(p);
129       p := NIL;
130   END;
131
132
133   // (pkt.11)
134
135   TYPE
136     TablicaInteger = array [0..5] of Integer;
137   VAR
138     a,b : Integer;
139     tab : TablicaInteger;
140     p: ^Integer;
141   BEGIN
142     tab[0] := 100;
143     tab[1] := 101;
144     tab[2] := 102;
145     tab[3] := 103;
146     tab[4] := 104;
147     tab[5] := 105;
148
149
150     WriteLn('tab[0]: ', tab[0] );
151     WriteLn('tab[1]: ', tab[1] );
152     WriteLn('tab[5]: ', tab[5] );
153
154     // p := @tab[0];
155     p := tab;
156     WriteLn('p^ : ', p^ );
157     WriteLn('(p+0)^: ', (p+0)^ );
158     WriteLn('(p+1)^: ', (p+1)^ );
159     WriteLn('(p+5)^: ', (p+5)^ );
160
161
162     ReadLn();
163   END.
164
165   // (pkt.12)
166   // GetMem(pointer,size);
167   // FreeMem(pointer,size);
168
169   CONST
170     LiczbaElementow = 5;
171   TYPE
```

```
172     TablicaInteger = array [0..LiczbaElementow-1] of Integer;
173   VAR
174     a,b,i : Integer;
175     tab : TablicaInteger;
176     p: ^Integer;
177   BEGIN
178     GetMem(p, sizeof(Integer) * LiczbaElementow); // Alokacja pamięci
179
180     for i:=0 to LiczbaElementow-1 do  p[i] := i+100;
181
182     for i:=0 to LiczbaElementow-1 do  WriteLn(p[i]);
183     WriteLn;
184
185     for i:=0 to LiczbaElementow-1 do  (p+i)^ := i+500;
186
187     for i:=0 to LiczbaElementow-1 do  WriteLn( (p+i)^ );
188     WriteLn;
189     for i:=0 to LiczbaElementow-1 do  WriteLn( p[i] );
190     WriteLn;
191
192     FreeMem(p, sizeof(Integer) * LiczbaElementow); // Zwolnienie pamięci
193
194     ReadLn();
195   END.
196
197   // (pkt.14-15)
198   VAR
199     osoba: Tosoba;
200     wskaznik_na_osobe: ^Tosoba;
201   Begin
202       new(wskaznik_na_osobe);
203
204       wskaznik_na_osobe^.imie := 'Jan';
205       WriteLn(wskaznik_na_osobe^.imie);
206
207       dispose(wskaznik_na_osobe);
208       wskaznik_na_osobe := NIL;
209   End.
210
211
212   // Realokacja:
213
214
215   program project_lab5_realokacja;
216   TYPE
217     Tosoba = record
218       imie: string[10];
219       nazwisko: string[10];
220     end;
221
222     NaszaTablicaDynamiczna = record
223       dane: ^Tosoba;
224       rozmiar: Integer;
225     end;
226
227   VAR
228     tablica : NaszaTablicaDynamiczna;
```

```
229     osoba: Tosoba;
230     wskaznik: ^Tosoba;
231     nowy_rozmiar, i:Integer;
232   BEGIN
233
234     osoba.imie:='Jan';
235     osoba.imie:='Kowalski';
236
237     // Przydzielenie pamięci   (2 osoby)
238     tablica.rozmiar := 2;
239     GetMem(tablica.dane, tablica.rozmiar * sizeof(Tosoba) );
240
241     tablica.dane[0] := osoba; // Wpisanie osoby (Jan Kowalski)
242     tablica.dane[1] := osoba; // (dwa razy)
243
244     // REALOKACJA \\
245
246     // alokacja nowej pamięci:
247     nowy_rozmiar := 5;
248     GetMem(wskaznik, nowy_rozmiar * sizeof(Tosoba) );
249
250     // przepisanie starych danych:
251     for i:=0 to tablica.rozmiar-1 do
252     begin
253       wskaznik[i] := tablica.dane[i];
254     end;
255
256     // zwolnienie starej pamięci:
257     FreeMem(tablica.dane,  tablica.rozmiar * sizeof(Tosoba) );
258
259     // zamiana danych na nowe:
260     tablica.dane := wskaznik;
261     tablica.rozmiar := nowy_rozmiar;
262
263     // ...
264
265     // zwolnienie  pamięci na koniec programu:
266     FreeMem(tablica.dane,  tablica.rozmiar * sizeof(Tosoba) );
267
268     ReadLn();
269   END.
270
271
272   // ######################################################
273   new(<ptr>)          // <ptr> = new <typ>
274   dispose(<ptr>)      // delete <ptr>
275
276   getmem (<ptr>,<size>*sizeof(<typ>))     // new <typ>[<size>]
277   freemem(<ptr>,<size>*sizeof(<typ>))     // delete [] <zm>
278
```