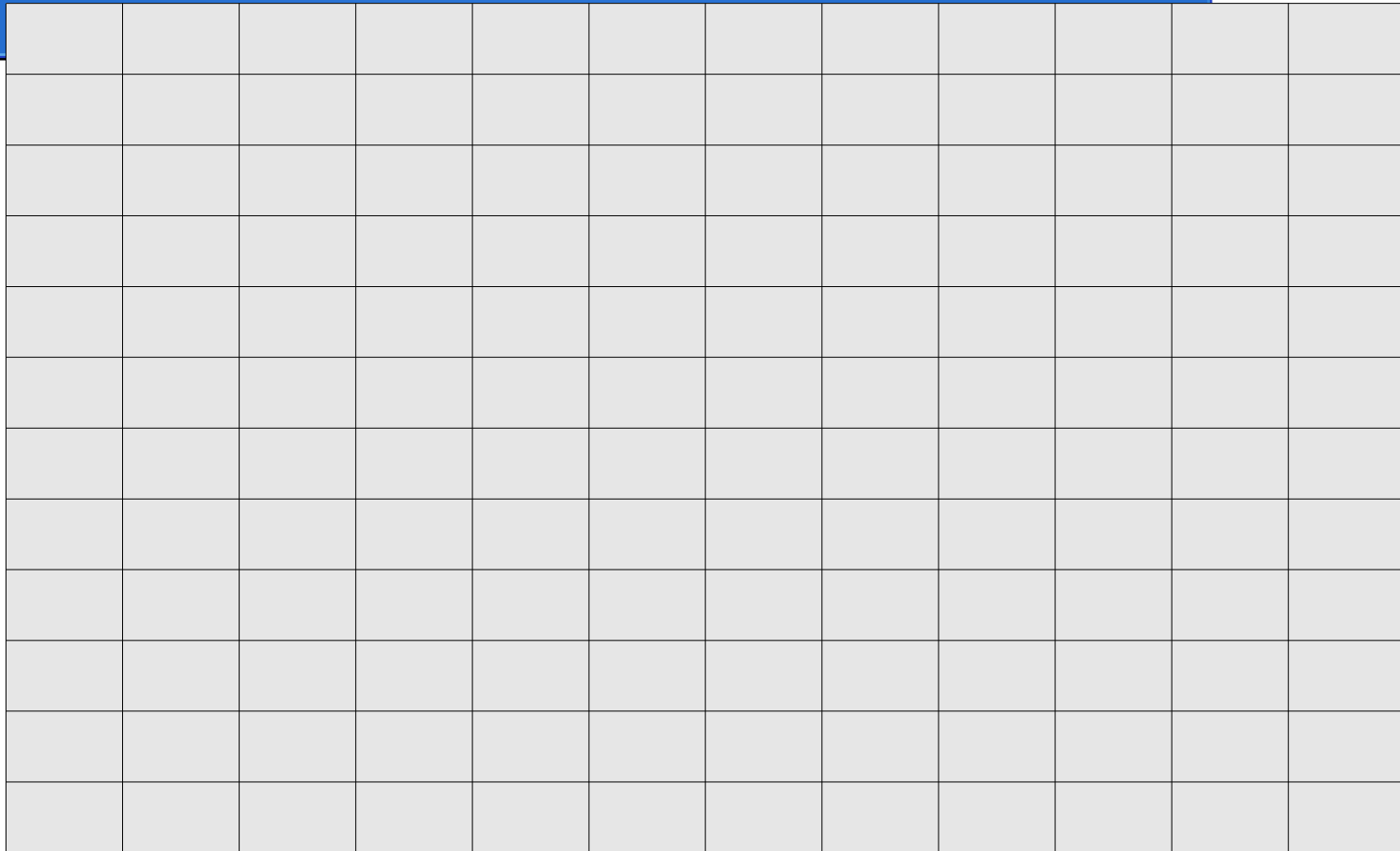


Reprezentare vizuală simplificată a memoriei heap și lucrului cu pointeri în limbajul C

Disclaimer

- Reprezentarea este simplistă și servește la vizualizarea mintală.
- Reprezentarea efectivă a valorilor în memorie pe octeți, cât și dimensiunea tipurilor de date (inclusiv a adreselor) depinde mult de arhitectura mașinii pe care se lucrează.

Să ne imaginăm că aceasta este o porțiune din memoria heap, unde au loc alocările dinamice.



Fiecare pătrățel din desen corespunde unui octet de memorie.

Fiecare octet are o adresă.

0x100	0x0101	0x0102	0x0103	0x0104	0x0105	0x0106	0x0107	0x0108	0x0109	0x010A	0x010B
0x010C	0x010D	0x010E	0x010F	0x0110	0x0111	0x0112	0x0113	0x0114	0x0115	0x0116	0x0117
0x0118	0x0119	0x011A	0x011B	0x011C	0x011D	0x011E	0x011F	0x0120	0x0121	0x0122	0x0123
0x0124	0x0125	0x0126	0x0127	0x0128	0x0129	0x012A	0x012B	0x012C	0x012D	0x012E	0x012F
0x0130	0x0131	0x0132	0x0133	0x0134	0x0135	0x0136	0x0137	0x0138	0x0139	0x013A	0x013B
0x013C	0x013D	0x013E	0x013F	0x0140	0x0141	0x0142	0x0143	0x0144	0x0145	0x0146	0x0147
0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Bineînțeles că fiecare pătrățel are **informație propriu-zisă**

0x0100	75	0x0101	41	0x0102	67	0x0103	42	0x0104	34	0x0105	24	0x0106	61	0x0107	11	0x0108	00	0x0109	00	0x010A	00	0x010B	00
0x010C	12	0x010D	5	0x010E	4f	0x010F	00	0x0110	00	0x0111	00	0x0112	00	0x0113	01	0x0114	00	0x0115	00	0x0116	14	0x0117	7b
0x0118	0b	0x0119	11	0x011A	00	0x011B	99	0x011C	5d	0x011D	12	0x011E	41	0x011F	2a	0x0120	00	0x0121	00	0x0122	42	0x0123	75
0x0124	12	0x0125	51	0x0126	2b	0x0127	2d	0x0128	77	0x0129	41	0x012A	21	0x012B	53	0x012C	00	0x012D	4a	0x012E	21	0x012F	4e
0x0130	57	0x0131	68	0x0132	25	0x0133	00	0x0134	00	0x0135	00	0x0136	00	0x0137	00	0x0138	00	0x0139	00	0x013A	00	0x013B	00
0x013C	04	0x013D	29	0x013E	74	0x013F	00	0x0140	88	0x0141	12	0x0142	9f	0x0143	24	0x0144	64	0x0145	62	0x0146	14	0x0147	64
0x0148	1d	0x0149	2f	0x014A	12	0x014B	41	0x014C	8e	0x014D	cf	0x014E	d4	0x014F	56	0x0150	32	0x0151	2b	0x0152	54	0x0153	4b
0x0154	54	0x0155	fd	0x0156	72	0x0157	42	0x0158	00	0x0159	2a	0x015A	e1	0x015B	18	0x015C	13	0x015D	4f	0x015E	2f	0x015F	bc
0x0160	1a	0x0161	e1	0x0162	75	0x0163	54	0x0164	00	0x0165	fc	0x0166	11	0x0167	9f	0x0168	54	0x0169	f2	0x016A	4e	0x016B	77
0x016C	b2	0x016D	1a	0x016E	d1	0x016F	1d	0x0170	bc	0x0171	4f	0x0172	24	0x0173	2a	0x0174	85	0x0175	4a	0x0176	12	0x0177	65
0x0178	35	0x0179	00	0x017A	21	0x017B	4f	0x017C	d4	0x017D	53	0x017E	52	0x017F	24	0x0180	00	0x0181	00	0x0182	00	0x0183	00
0x0184	64	0x0185	00	0x0186	0d	0x0187	5a	0x0188	e1	0x0189	41	0x018A	7b	0x018B	66	0x018C	5d	0x018D	74	0x018E	83	0x018F	21

Programele cer sistemului de operare să le **aloc**e octeți.

Program 1
Program 2
Program 3

0x0100	75	0x0101	41	0x0102	67	0x0103	42	0x0104	34	0x0105	24	0x0106	61	0x0107	11	0x0108	00	0x0109	00	0x010A	00	0x010B	00
0x010C	12	0x010D	5	0x010E	4f	0x010F	00	0x0110	00	0x0111	00	0x0112	00	0x0113	01	0x0114	00	0x0115	00	0x0116	14	0x0117	7b
0x0118	0b	0x0119	11	0x011A	00	0x011B	99	0x011C	5d	0x011D	12	0x011E	41	0x011F	2a	0x0120	00	0x0121	00	0x0122	42	0x0123	75
0x0124	12	0x0125	51	0x0126	2b	0x0127	2d	0x0128	77	0x0129	41	0x012A	21	0x012B	53	0x012C	00	0x012D	4a	0x012E	21	0x012F	4e
0x0130	57	0x0131	68	0x0132	25	0x0133	00	0x0134	00	0x0135	00	0x0136	00	0x0137	00	0x0138	00	0x0139	00	0x013A	00	0x013B	00
0x013C	04	0x013D	29	0x013E	74	0x013F	00	0x0140	88	0x0141	12	0x0142	9f	0x0143	24	0x0144	64	0x0145	62	0x0146	14	0x0147	64
0x0148	1d	0x0149	2f	0x014A	12	0x014B	41	0x014C	8e	0x014D	cf	0x014E	d4	0x014F	56	0x0150	32	0x0151	2b	0x0152	54	0x0153	4b
0x0154	54	0x0155	fd	0x0156	72	0x0157	42	0x0158	00	0x0159	2a	0x015A	e1	0x015B	18	0x015C	13	0x015D	4f	0x015E	2f	0x015F	bc
0x0160	1a	0x0161	e1	0x0162	75	0x0163	54	0x0164	00	0x0165	fc	0x0166	11	0x0167	9f	0x0168	54	0x0169	f2	0x016A	4e	0x016B	77
0x016C	b2	0x016D	1a	0x016E	d1	0x016F	1d	0x0170	bc	0x0171	4f	0x0172	24	0x0173	2a	0x0174	85	0x0175	4a	0x0176	12	0x0177	65
0x0178	35	0x0179	00	0x017A	21	0x017B	4f	0x017C	d4	0x017D	53	0x017E	52	0x017F	24	0x0180	00	0x0181	00	0x0182	00	0x0183	00
0x0184	64	0x0185	00	0x0186	0d	0x0187	5a	0x0188	e1	0x0189	41	0x018A	7b	0x018B	66	0x018C	5d	0x018D	74	0x018E	83	0x018F	21

Memoria alocată unui program nu trebuie să fie neapărat contiguă.

0x0148 1d	0x0149 2f	0x014A 12	0x014B 41	0x014C 8e	0x014D cf	0x014E d4	0x014F 56	0x0150 32	0x0151 2b	0x0152 54	0x0153 4b
0x0154 54	0x0155 fd	0x0156 72	0x0157 42	0x0158 00	0x0159 2a	0x015A e1	0x015B 18	0x015C 13	0x015D 4f	0x015E 2f	0x015F bc
0x0160 1a	0x0161 e1	0x0162 75	0x0163 54	0x0164 00	0x0165 fc	0x0166 11	0x0167 9f	0x0168 54	0x0169 f2	0x016A 4e	0x016B 77
0x016C b2	0x016D 1a	0x016E d1	0x016F 1d	0x0170 bc	0x0171 4f	0x0172 24	0x0173 2a	0x0174 85	0x0175 4a	0x0176 12	0x0177 65
0x0178 35	0x0179 00	0x017A 21	0x017B 4f	0x017C d4	0x017D 53	0x017E 52	0x017F 24	0x0180 00	0x0181 00	0x0182 00	0x0183 00
0x0184 64	0x0185 00	0x0186 0d	0x0187 5a	0x0188 e1	0x0189 41	0x018A 7b	0x018B 66	0x018C 5d	0x018D 74	0x018E 83	0x018F 21

În octeții alocăți, programele pot stoca orice fel de informații.

Întregul pe 2 octeți 002a = 42

1d	2f	12	41	8e	cf	d4	56	32	2b	54	4b
54	fd	72	42	00	2a	e1	18	13	4f	2f	bc
1a	e1	75	54	00	fc	11	9f	54	f2	4e	77
b2	1a	d1	1d	bc	4f	24	2a	85	4a	12	65
35	00	21	4f	d4	53	52	24	00	00	00	00
64	00	0d	5a	e1	41	7b	66	5d	74	83	21

Întregul pe 1 octet 77, echivalent cu caracterul ASCII 'w'

Un pointer este o variabilă care stochează o adresă (de obicei, adresa unei alte variabile).

0x0B12 01	0x0B13 6A
--------------	--------------

0x0148 1d	0x0149 2f	0x014A 12	0x014B 41	0x014C 8e	0x014D cf	0x014E d4	0x014F 56	0x0150 32	0x0151 2b	0x0152 54	0x0153 4b
0x0154 54	0x0155 fd	0x0156 72	0x0157 42	0x0158 00	0x0159 2a	0x015A e1	0x015B 18	0x015C 13	0x015D 4f	0x015E 2f	0x015F bc
0x0160 1a	0x0161 e1	0x0162 75	0x0163 54	0x0164 00	0x0165 fc	0x0166 11	0x0167 9f	0x0168 54	0x0169 f2	0x016A 4e	0x016B 77
0x016C b2	0x016D 1a	0x016E d1	0x016F 1d	0x0170 bc	0x0171 4f	0x0172 24	0x0173 2a	0x0174 85	0x0175 4a	0x0176 12	0x0177 65
0x0178 35	0x0179 00	0x017A 21	0x017B 4f	0x017C d4	0x017D 53	0x017E 52	0x017F 24	0x0180 00	0x0181 00	0x0182 00	0x0183 00
0x0184 64	0x0185 00	0x0186 0d	0x0187 5a	0x0188 e1	0x0189 41	0x018A 7b	0x018B 66	0x018C 5d	0x018D 74	0x018E 83	0x018F 21

De exemplu, ar putea fi un pointer către un întreg pe 2 octeți aflat la adresa 0x016A, având tipul `int*`.

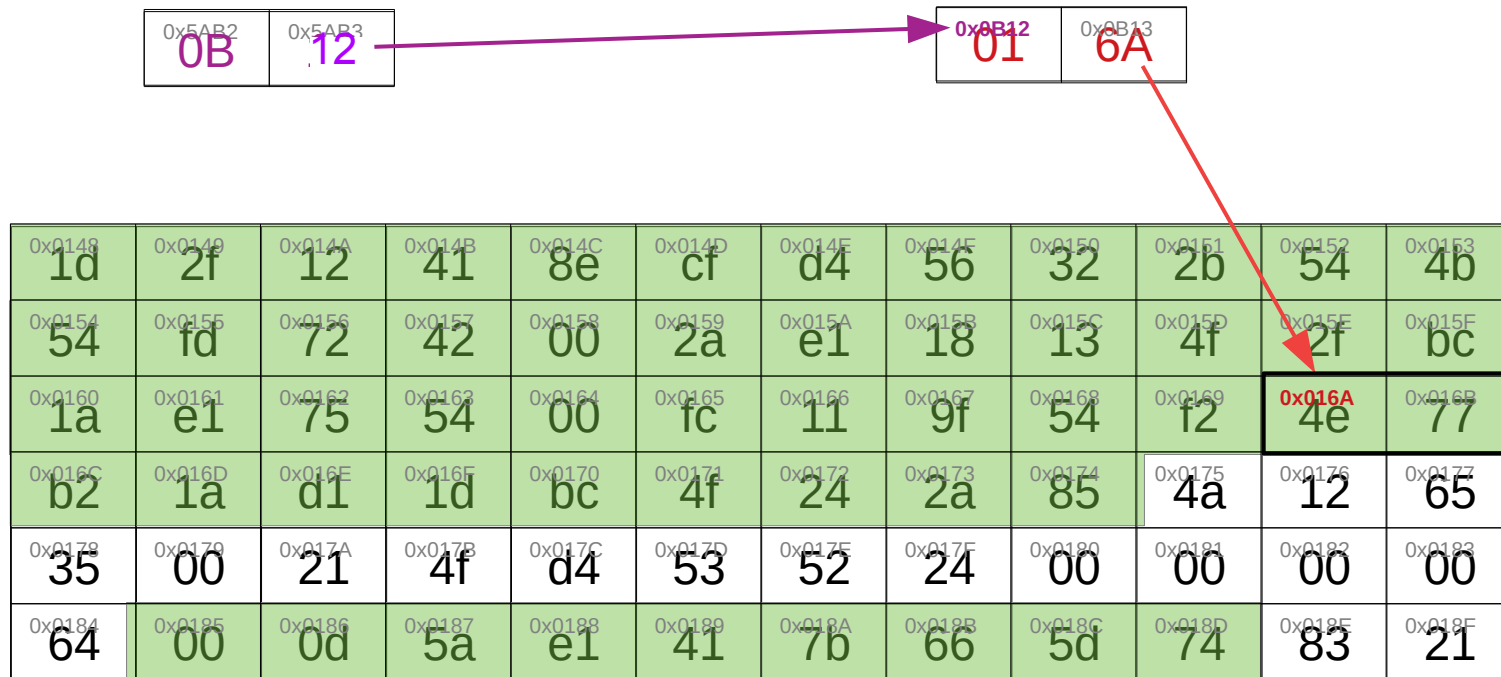
0x0B12 01	0x0B13 6A
--------------	--------------

0x0148 1d	0x0149 2f	0x014A 12	0x014B 41	0x014C 8e	0x014D cf	0x014E d4	0x014F 56	0x0150 32	0x0151 2b	0x0152 54	0x0153 4b
0x0154 54	0x0155 fd	0x0156 72	0x0157 42	0x0158 00	0x0159 2a	0x015A e1	0x015B 18	0x015C 13	0x015D 4f	0x015E 2f	0x015F bc
0x0160 1a	0x0161 e1	0x0162 75	0x0163 54	0x0164 00	0x0165 fc	0x0166 11	0x0167 9f	0x0168 54	0x0169 f2	0x016A 4e	0x016B 77
0x016C b2	0x016D 1a	0x016E d1	0x016F 1d	0x0170 bc	0x0171 4f	0x0172 24	0x0173 2a	0x0174 85	0x0175 4a	0x0176 12	0x0177 65
0x0178 35	0x0179 00	0x017A 21	0x017B 4f	0x017C d4	0x017D 53	0x017E 52	0x017F 24	0x0180 00	0x0181 00	0x0182 00	0x0183 00
0x0184 64	0x0185 00	0x0186 0d	0x0187 5a	0x0188 e1	0x0189 41	0x018A 7b	0x018B 66	0x018C 5d	0x018D 74	0x018E 83	0x018F 21

Dar pointerul are și el, ca orice informație din memorie, o adresă...

Și ar putea exista o altă variabilă care să conțină adresa lui: un pointer la pointer:

`int**`



Operația de alocare

```
int* a;  
a = (int*)malloc(sizeof(int) * 5);
```

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de alocare

```
→ int* a;  
  a = (int*)malloc(sizeof(int) * 5);
```

conținutul efectiv al pointerului a

0x0B12 ??	0x0B13 ??
--------------	--------------

Se alocă memorie static (pe stivă) pentru variabila-pointer a.
Inițial are o valoare necunoscută. („gunoi”)

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de alocare

```
→ int* a;  
a = (int*)malloc(sizeof(int) * 5);
```

conținutul efectiv al pointerului a

0x0B12 ??	0x0B13 ??
--------------	--------------

Sistemul de operare găsește și atribuie programului un bloc de memorie de dimensiunea cerută. Programul îl poate folosi acum pentru a reține informații proprii.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de alocare

```
→ int* a;  
a = (int*)malloc(sizeof(int) * 5);
```

conținutul efectiv al pointerului a

0x0B12 ??	0x0B13 ??
--------------	--------------

Presupunem, pentru simplitate, că sizeof(int) este 2 iar blocul alocat de SO este ca în figura de mai jos.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de alocare

```
int* a;  
→ a = (int*)malloc(sizeof(int) * 5);
```

conținutul efectiv al pointerului a

0x0B12 ??	0x0B13 ??
--------------	--------------

În acest bloc încap 5 întregi cu dimensiunea de 2 octeți.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de alocare

```
int* a;  
→ a = (int*)malloc(sizeof(int) * 5);
```

conținutul efectiv al pointerului a

0x0B12 ??	0x0B13 ??
--------------	--------------

După ce alocarea a avut loc, adresa de început a blocului (**vectorului**) este returnată de funcția malloc și stocată în pointerul **a**.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de alocare

```
a[0] = 0;  
a[1] = 1;  
a[3] = 12;
```

conținutul efectiv al pointerului a

0x0B12 01	0x0B13 6E
--------------	--------------

a[0], a[1], a[2] etc sunt referințe directe la memoria alocată. Lucrul cu ele duce la aceesarea / modificarea efectivă a valorilor de la adresele respective.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E 00	0x016F 00	0x0170 00	0x0171 01	0x0172	0x0173	0x0174 00	0x0175 0d	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de dealocare

```
free(a);
```

conținutul efectiv al pointerului a

0x0B12 01	0x0B13 6E
--------------	--------------

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E 00	0x016F 00	0x0170 00	0x0171 01	0x0172	0x0173	0x0174 00	0x0175 0d	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de dealocare

→ free(a);

conținutul efectiv al pointerului a

0x0B12 01	0x0B13 6E
--------------	--------------

Blocul de memorie care începe la adresa indicată de pointerul a (0x016E) devine memorie liberă. Din acest moment, alte aplicații ar putea primi acea zonă de memorie.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E 00	0x016F 00	0x0170 00	0x0171 01	0x0172	0x0173	0x0174 00	0x0175 0d	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de dealocare

→ free(a);

conținutul efectiv al pointerului a

0x0B12 01	0x0B13 6E
--------------	--------------

Eliberarea memoriei nu „curăță” informația lăsată acolo.

Totuși, încercarea de a accesa pointerul după dealocarea memoriei va duce la o eroare catastrofală, deoarece programul încearcă să acceseze memorie ce nu-i aparține.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E 00	0x016F 00	0x0170 00	0x0171 01	0x0172	0x0173	0x0174 00	0x0175 0d	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Operația de dealocare

`free(a);`

`a = 0;`

conținutul efectiv al pointerului a

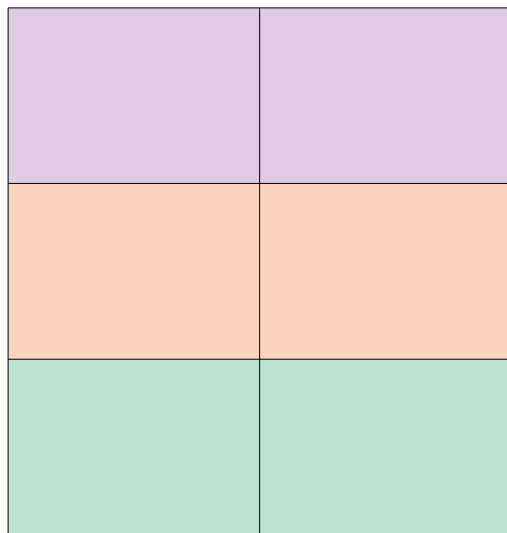
0x0B12 00	0x0B13 00
--------------	--------------

De aceea, o bună practică de „igienă” a programării (adesea necesară) este să marcăm pointerul cu valoarea zero.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E 00	0x016F 00	0x0170 00	0x0171 01	0x0172	0x0173	0x0174 00	0x0175 0d	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

Presupunem că vrem să alocăm spațiu pentru o matrice cu 3 rânduri și 2 coloane, precum cea de mai jos:



A diagram of a 3x2 matrix. It consists of a 3x2 grid of squares. The top row has two light purple squares. The middle row has two light orange squares. The bottom row has two light green squares. The squares are arranged in a 3x2 grid, with a vertical line separating the two columns and a horizontal line separating the three rows.

Alocarea dinamică a unei matrici

```
int** T = (int**)malloc(sizeof(int*) * 3);
```

Începem cu alocarea unui „vector de pointeri”, câte unul pentru fiecare linie.
Presupunem că `sizeof(int*)` este 2.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

```
int** T = (int**)malloc(sizeof(int*) * 3);
```

conținutul efectiv al pointerului T

0x0B12 01	0x0B13 48
---------------------	---------------------

Începem cu alocarea unui „vector de pointeri”, câte unul pentru fiecare linie.
Presupunem că sizeof(int*) este 2.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

```
int** T = (int**)malloc(sizeof(int*) * 3);
```

conținutul efectiv al pointerului T

0x0B12 01	0x0B13 48
---------------------	---------------------

Apoi alocăm, pentru fiecare din cele 3 rânduri, spațiu cât pentru 2 variabile int.

0x0148	0x0149	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

```
T[0] = (int*)malloc(sizeof(int) * 2);
```

conținutul efectiv al pointerului T

0x0B12 01	0x0B13 48
---------------------	---------------------

Primul rând...

0x0148 01	0x0149 62	0x014A	0x014B	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

```
T[1] = (int*)malloc(sizeof(int) * 2);
```

conținutul efectiv al pointerului T

0x0B12 01	0x0B13 48
---------------------	---------------------

Al doilea rând...

0x0148 01	0x0149 62	0x014A 01	0x014B 68	0x014C	0x014D	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

```
T[2] = (int*)malloc(sizeof(int) * 2);
```

conținutul efectiv al pointerului T

0x0B12 01	0x0B13 48
---------------------	---------------------

Al treilea rând...

0x0148 01	0x0149 62	0x014A 01	0x014B 68	0x014C 01	0x014D 70	0x014E	0x014F	0x0150	0x0151	0x0152	0x0153
0x0154	0x0155	0x0156	0x0157	0x0158	0x0159	0x015A	0x015B	0x015C	0x015D	0x015E	0x015F
0x0160	0x0161	0x0162	0x0163	0x0164	0x0165	0x0166	0x0167	0x0168	0x0169	0x016A	0x016B
0x016C	0x016D	0x016E	0x016F	0x0170	0x0171	0x0172	0x0173	0x0174	0x0175	0x0176	0x0177
0x0178	0x0179	0x017A	0x017B	0x017C	0x017D	0x017E	0x017F	0x0180	0x0181	0x0182	0x0183
0x0184	0x0185	0x0186	0x0187	0x0188	0x0189	0x018A	0x018B	0x018C	0x018D	0x018E	0x018F

Alocarea dinamică a unei matrici

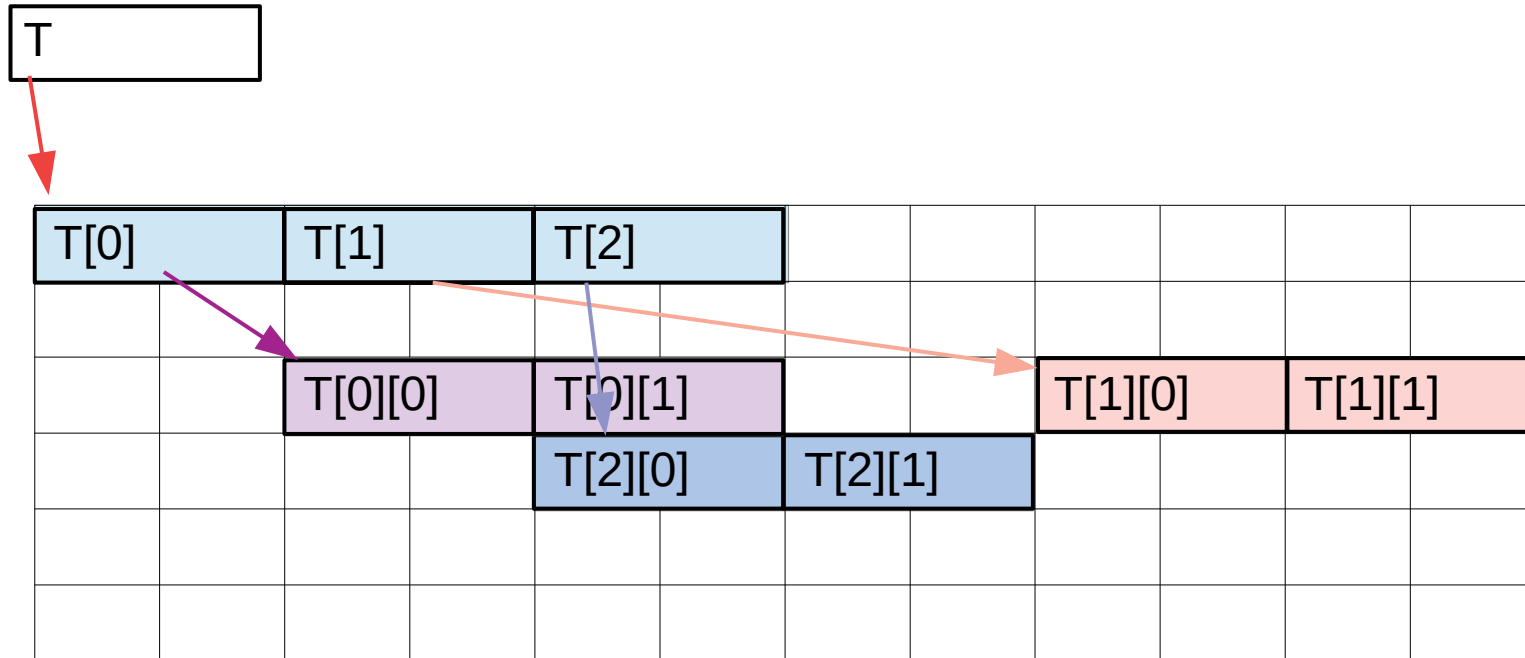
Împreună, aceste rânduri formează matricea



$\tau[0][0]$	$\tau[0][1]$
$\tau[1][0]$	$\tau[1][1]$
$\tau[2][0]$	$\tau[2][1]$

The diagram illustrates the mapping of a 2D array T to a 1D array. The 2D array T is represented as a grid of cells. The 1D array is represented as a row of cells. Arrows indicate the mapping from 2D to 1D: $T[0]$ maps to $T[0][0]$, $T[1]$ maps to $T[0][1]$, and $T[2]$ maps to $T[1][0]$.

Dealocarea dinamică a unei matrici



Dealocarea dinamică a unei matrici

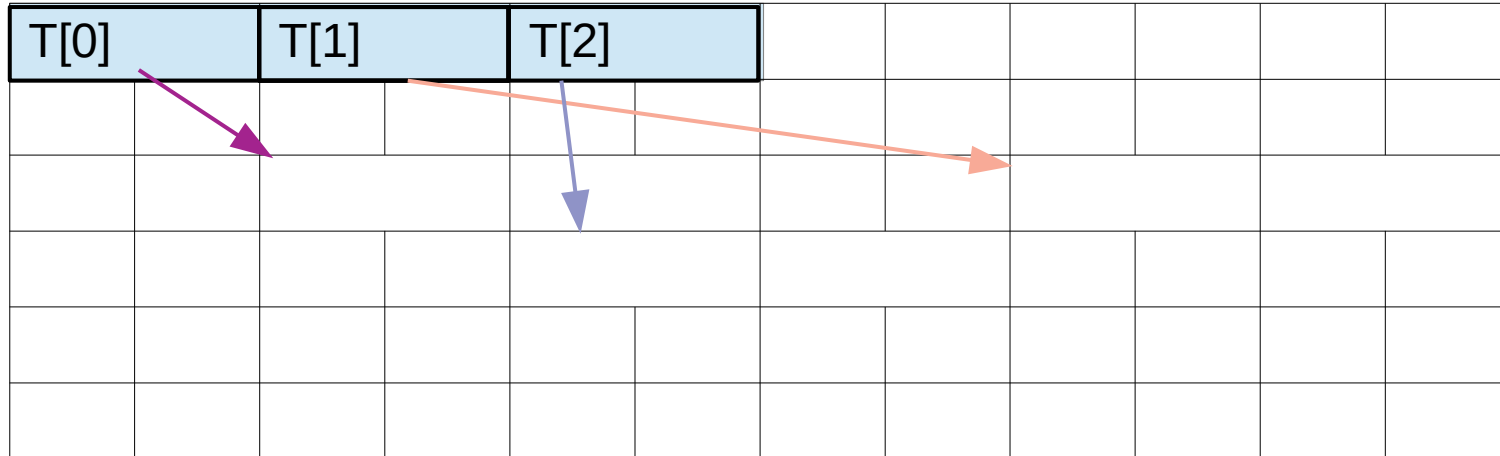
Se dealocă rândurile, unul câte unul:

```
free(T[0]);
```

```
free(T[1]);
```

```
free(T[2]);
```

T



Dealocarea dinamică a unei matrici

Apoi se dealocă lista de pointeri la rânduri:

```
free(T);
```

