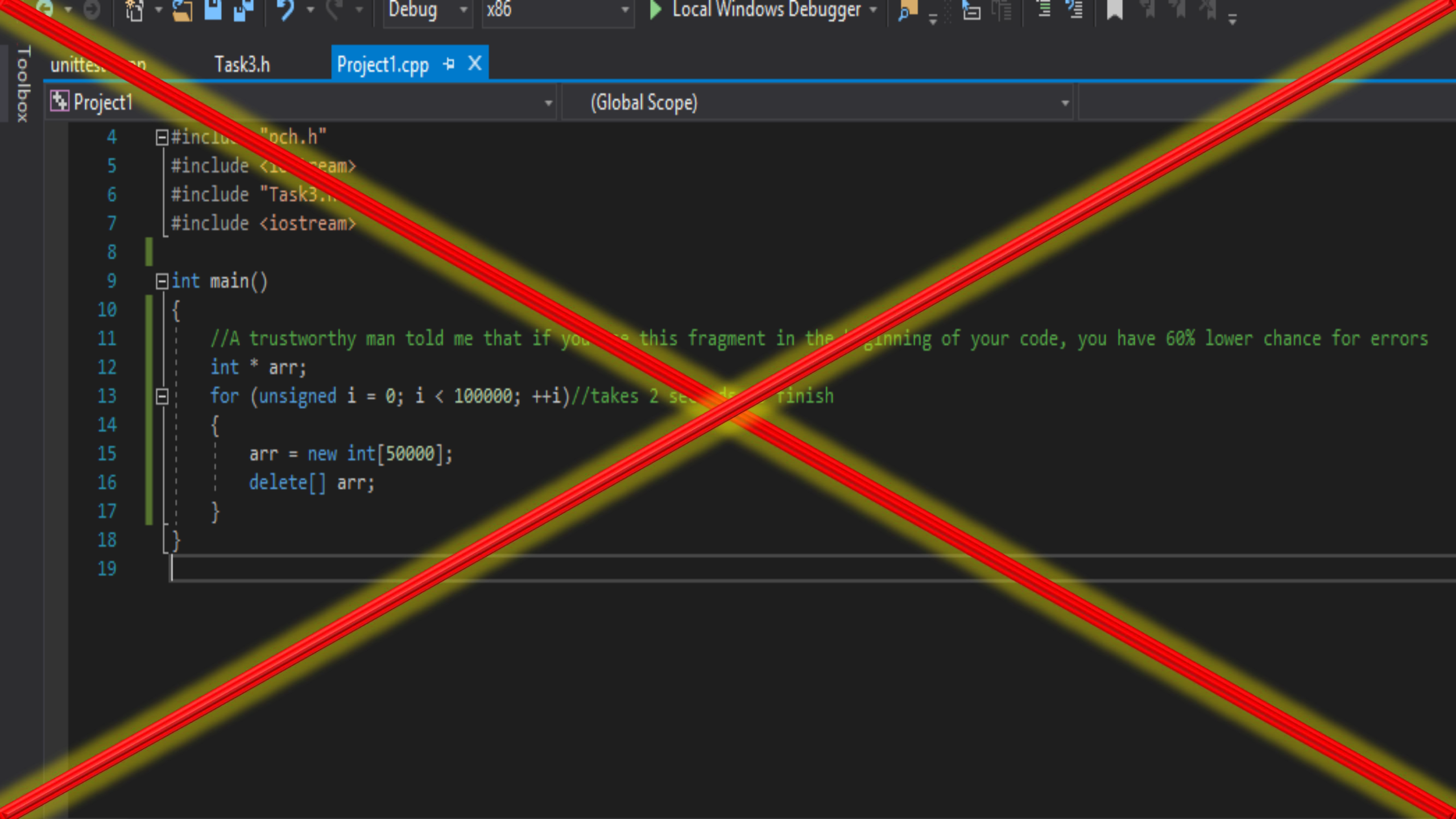
Работа с динамична памет

- Работата с динамична памет включва:

1. Заделяне на такава памет
2. Обработка на данни
3. Освобождаване на заделената памет

- Като цяло 2. е optional, но реално ние използваме динамична памет точно заради 2.


Debug
x86
Local Windows Debugger
Task3.h
Project1.cpp
Project1
(Global Scope)
#include "och.h"
#include <iostream>
#include "Task3.h"
#include <iostream>
int main()
{
//A trustworthy man told me that if you use this fragment in the beginning of your code, you have 60% lower chance for errors
int * arr;
for (unsigned i = 0; i < 100000; ++i)//takes 2 seconds to finish
{
arr = new int[50000];
delete[] arr;
}
}

Заделяне на динамична памет

- За заделяне на динамична памет се използват операторите:
  - new <тип> [(<стойност>)] – заделя памет за точно един нов обект, инициализира го и връща пойнтър към него
  - new <тип>[<число n>] – заделя памет за n-мерна редица, инициализира всички обекти в нея и връща пойнтър към първия
- Примери:
  - char * dChar = new char;
  - char * dCharA = new char('A');
  - char * dCharArr = new char[6];

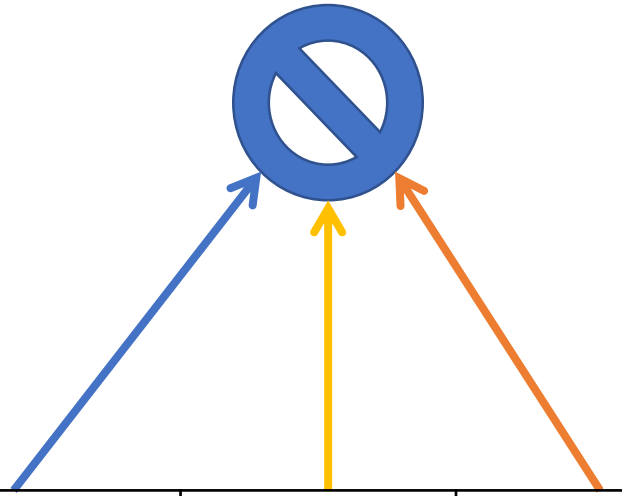
Заделяне на динамична памет - визуализация

```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB
0xC	0xD	0xE	0xF	0x10	0x11
0x12	0x13	0x14	0x15	0x16	0x17

Заделяне на динамична памет - визуализация

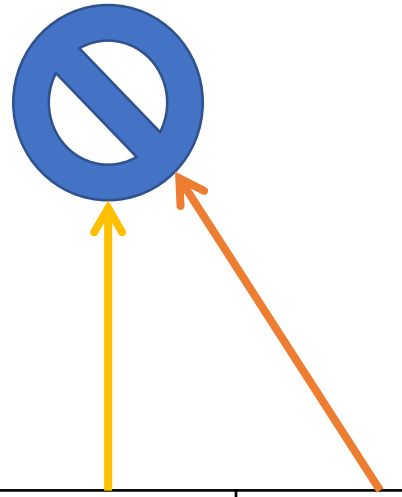
```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
```



0x0	0x1	0x2	0x3-dChar	0x4-dCharA	0x5-dCharArr
			undefined	undefined	undefined
0x6	0x7	0x8	0x9	0xA	0xB
0xC	0xD	0xE	0xF	0x10	0x11
0x12	0x13	0x14	0x15	0x16	0x17

Заделяне на динамична памет - визуализация

```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
```



0x0	0x1	0x2	0x3-dChar	0x4-dCharA	0x5-dCharArr
			0x8	undefined	undefined
0x6	0x7	0x8	0x9	0xA	0xB
		undefined			
0xC	0xD	0xE	0xF	0x10	0x11
0x12	0x13	0x14	0x15	0x16	0x17

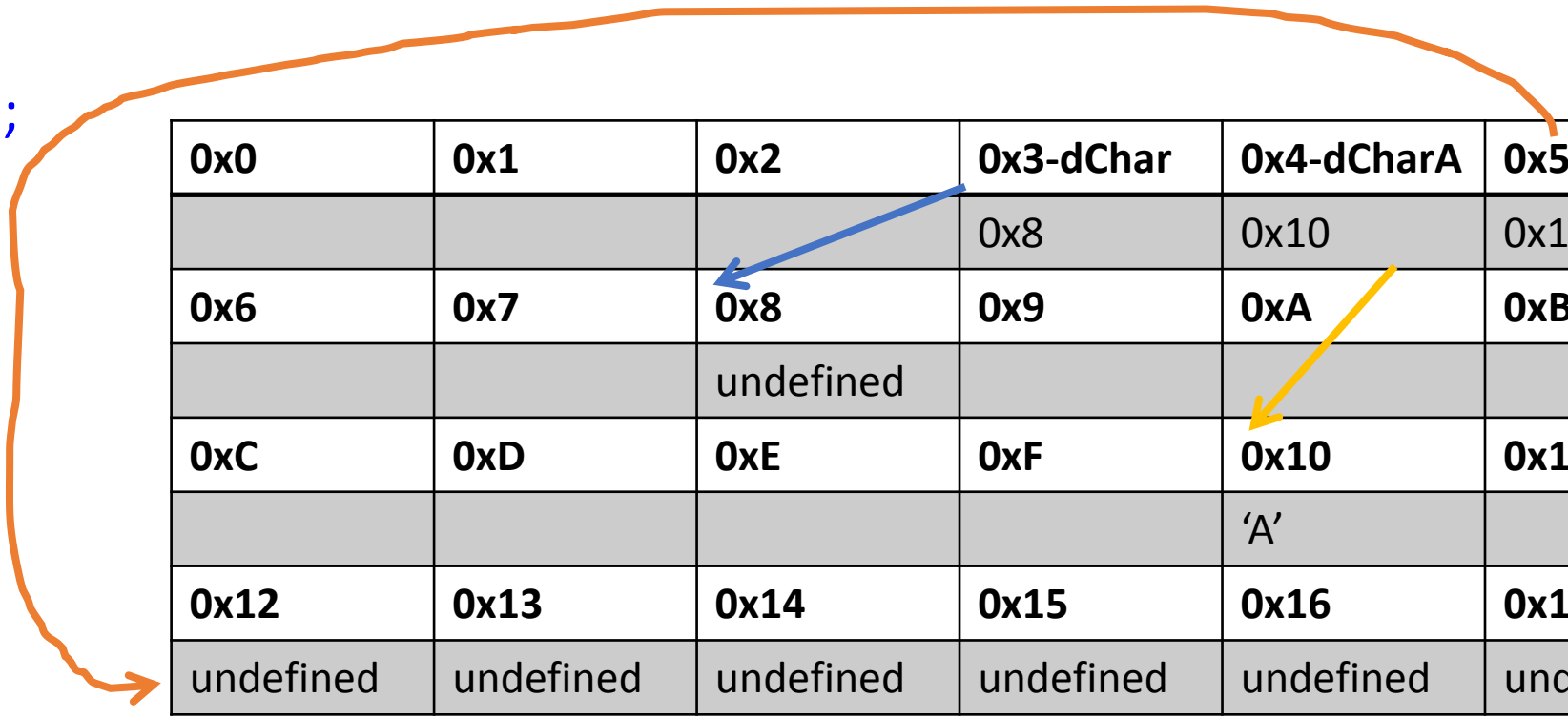
Заделяне на динамична памет - визуализация

```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
```

0x0	**0x1**	**0x2**	**0x3-dChar**	**0x4-dCharA**	**0x5-dCharArr**
			0x8	0x10	undefined
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB**
		undefined			
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
				'A'	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**

Заделяне на динамична памет - визуализация

```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
```



0x0	**0x1**	**0x2**	**0x3-dChar**	**0x4-dCharA**	**0x5-dCharArr**
			0x8	0x10	0x12
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB**
		undefined			
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
				'A'	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
undefined	undefined	undefined	undefined	undefined	undefined

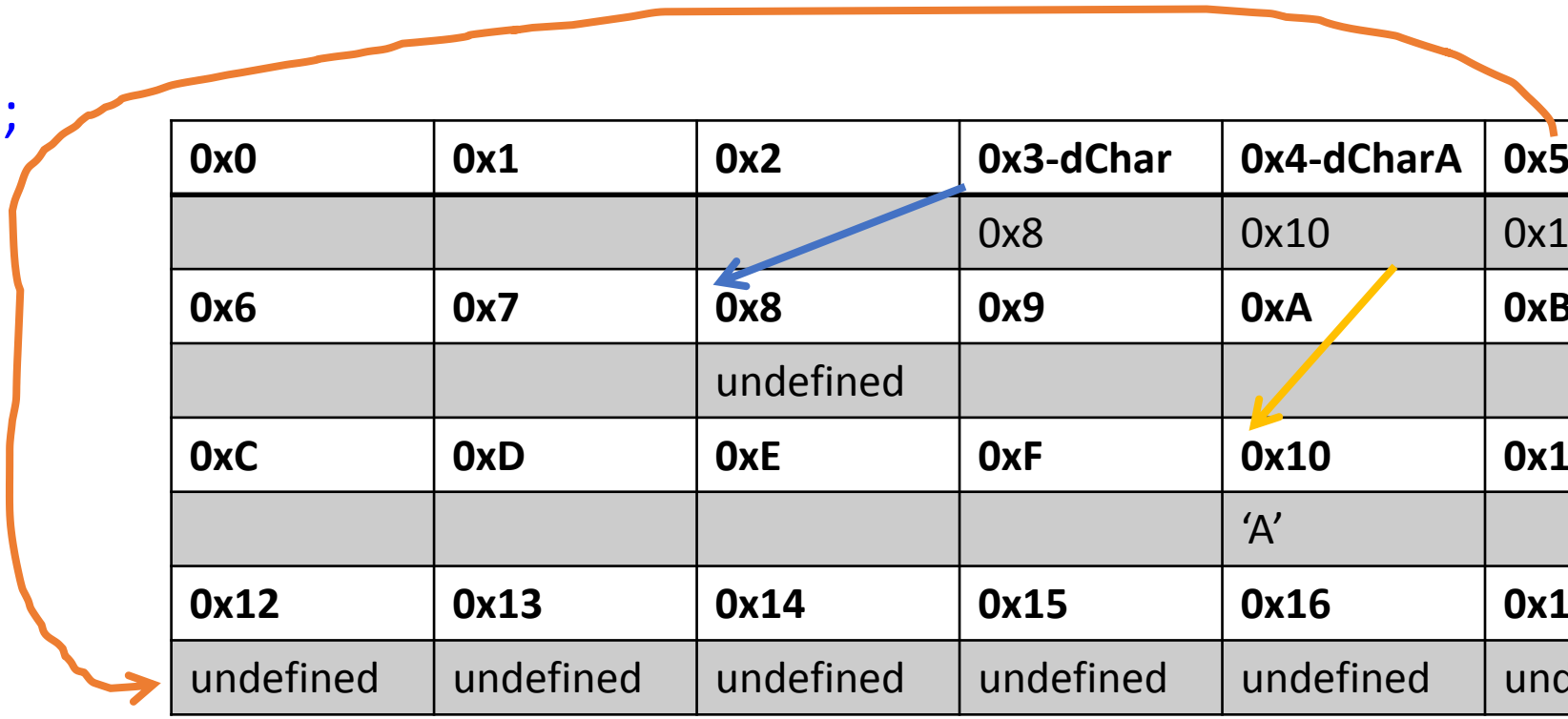
Освобождаване на заделена памет

- За освобождаване на динамична памет се използват операторите:
  - delete <адрес>
  - delete[] <адрес>

- Примери(спрямо предишните примери):
  - delete dChar;
  - delete dCharA;
  - delete[] dCharArr;

Освобождаване на заделена памет - визуализация

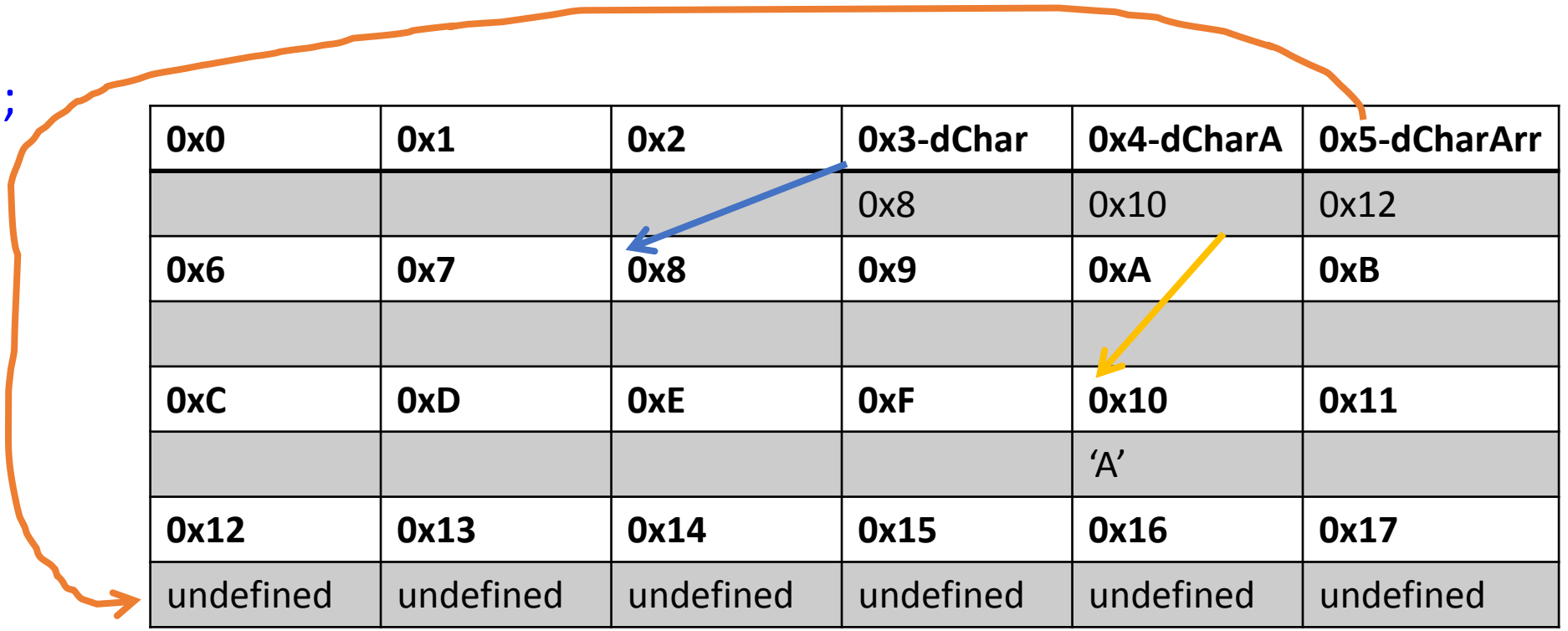
```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
delete dChar;
delete dCharA;
delete[] dCharArr;
```



0x0	**0x1**	**0x2**	**0x3-dChar**	**0x4-dCharA**	**0x5-dCharArr**
			0x8	0x10	0x12
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB**
		undefined			
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
				'A'	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
undefined	undefined	undefined	undefined	undefined	undefined

Освобождаване на заделена памет - визуализация

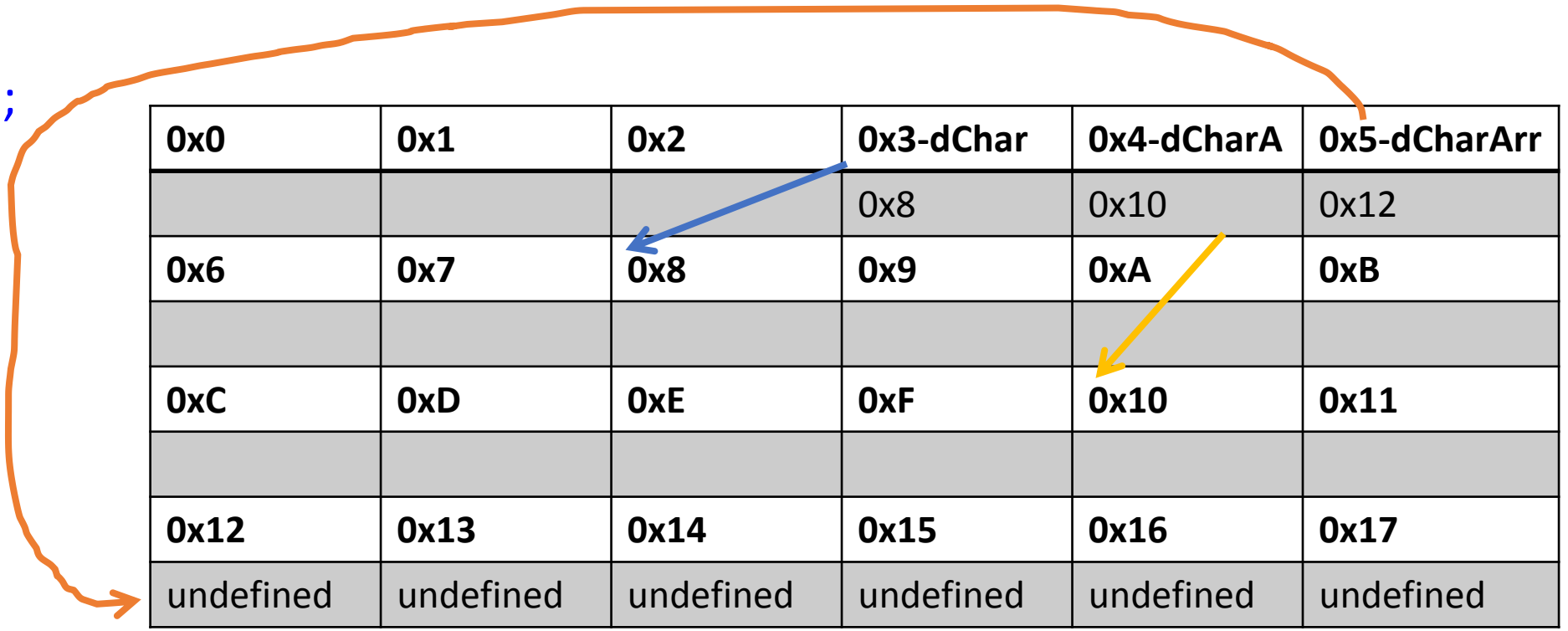
```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
delete dChar;
delete dCharA;
delete[] dCharArr;
```



0x0	**0x1**	**0x2**	**0x3-dChar**	**0x4-dCharA**	**0x5-dCharArr**
			0x8	0x10	0x12
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB**
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
				'A'	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
undefined	undefined	undefined	undefined	undefined	undefined

Освобождаване на заделена памет - визуализация

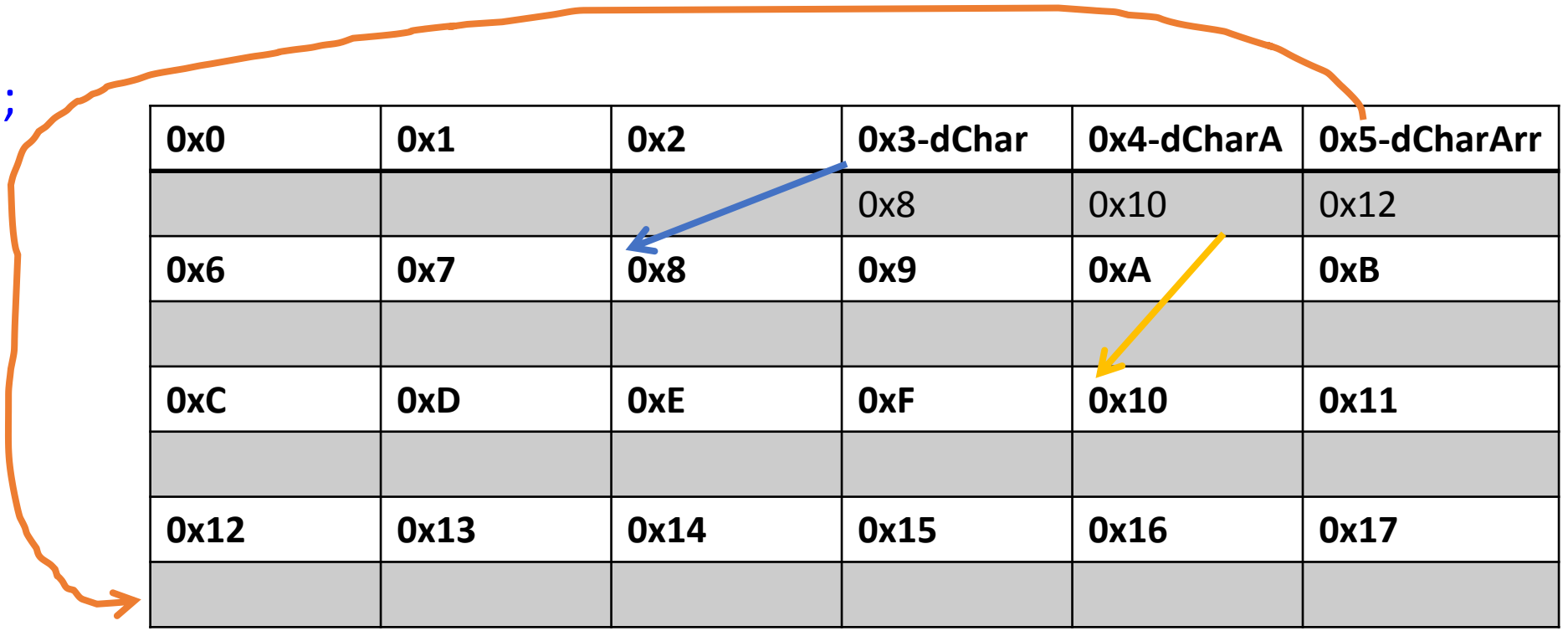
```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
delete dChar;
delete dCharA;
delete[] dCharArr;
```



0x0	**0x1**	**0x2**	**0x3-dChar**	**0x4-dCharA**	**0x5-dCharArr**
			0x8	0x10	0x12
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB**
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
undefined	undefined	undefined	undefined	undefined	undefined

Освобождаване на заделена памет - визуализация

```
char * dChar, * dCharA, * dCharArr;
dChar = new char;
dCharA = new char('A');
dCharArr = new char[6];
delete dChar;
delete dCharA;
delete[] dCharArr;
```



0x0	0x1	0x2	0x3-dChar	0x4-dCharA	0x5-dCharArr
			0x8	0x10	0x12
0x6	0x7	0x8	0x9	0xA	0xB
0xC	0xD	0xE	0xF	0x10	0x11
0x12	0x13	0x14	0x15	0x16	0x17

Освобождаване на заделена памет

- delete операторите могат да освобождават само динамично заделена памет
- Не е позволено освобождаването на стекова памет:

```
int a;
int* b = &a;
delete b;
```

- Не е позволено частично освобождаване на памет:

```
int* a = new int[10];
int * b = a+1;
delete[] b;
```

Освобождаване на заделена памет

- За ваше улеснение, може да използвате правилото:
  - new => delete
  - new[] => delete[] //delete[] си знае колко памет трябва да освободи
- След освобождаването на дадена памет, тя става недостъпна и обръщането към нея обикновено води до фойерверки
- delete и delete[] върху nullptr са безобидни и не правят нищо

- Винаги освобождавайте паметта, която сте заделили!!!

Освобождаване на заделена памет

- standard (5.3.5/2) :
  - In the first alternative (delete object), the value of the operand of delete shall be a pointer to a non-array object or a pointer to a sub-object (1.8) representing a base class of such an object (clause 10). **If not, the behavior is undefined.**
  - In the second alternative (delete array), the value of the operand of delete shall be the pointer value which resulted from a previous array new-expression. **If not, the behavior is undefined.**

Обработка на динамични данни

- Особеностите на динамичните данни са:
  - заделяне на памет и освобождаването ѝ се извършва по време на изпълнение
  - липса на име

- Всичко останало си е както и преди, като трябва:
  - да се съобрази, че се обръщаме към пойнтър от дадения тип, а не просто към обект
  - от сходствата между пойнтъри и масиви следва, че можем спокойно да използваме оператор [] //припомнете си какво разгледахме там

Обработка на динамични данни

- Възможността да контролираме кога дадена памет да бъде освободена ни дава изненадващо много нови възможности
- Вече функция, връщаща пойнтър, може да има много повече приложения от преди
- With Great Power Comes Great Responsibility!

Обработка на динамични данни

- Как се създава матрица NxM ?

```
int ** Matrix (const unsigned n, const unsigned m)
{
        int ** tmp = new int * [n];
        for(unsigned i = 0; i<n; ++i)
                tmp[i] = new int [m];
        return tmp;
}
```

Обработка на динамични данни

- Как се изтрива матрица NxM ?

```
int ** A = Matrix(3,2);
for(unsigned i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB
0xC	0xD	0xE	0xF	0x10	0x11
0x12	0x13	0x14	0x15	0x16	0x17

Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```

0x0 - n	0x1	0x2	0x3	0x4 - m	0x5
			11		
0x6	0x7	0x8	0x9	0xA	0xB
	10				
0xC	0xD	0xE	0xF	0x10	0x11
0x12	0x13	0x14	0x15	0x16	0x17

Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```

0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
		Undefined	Undefined	Undefined	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**

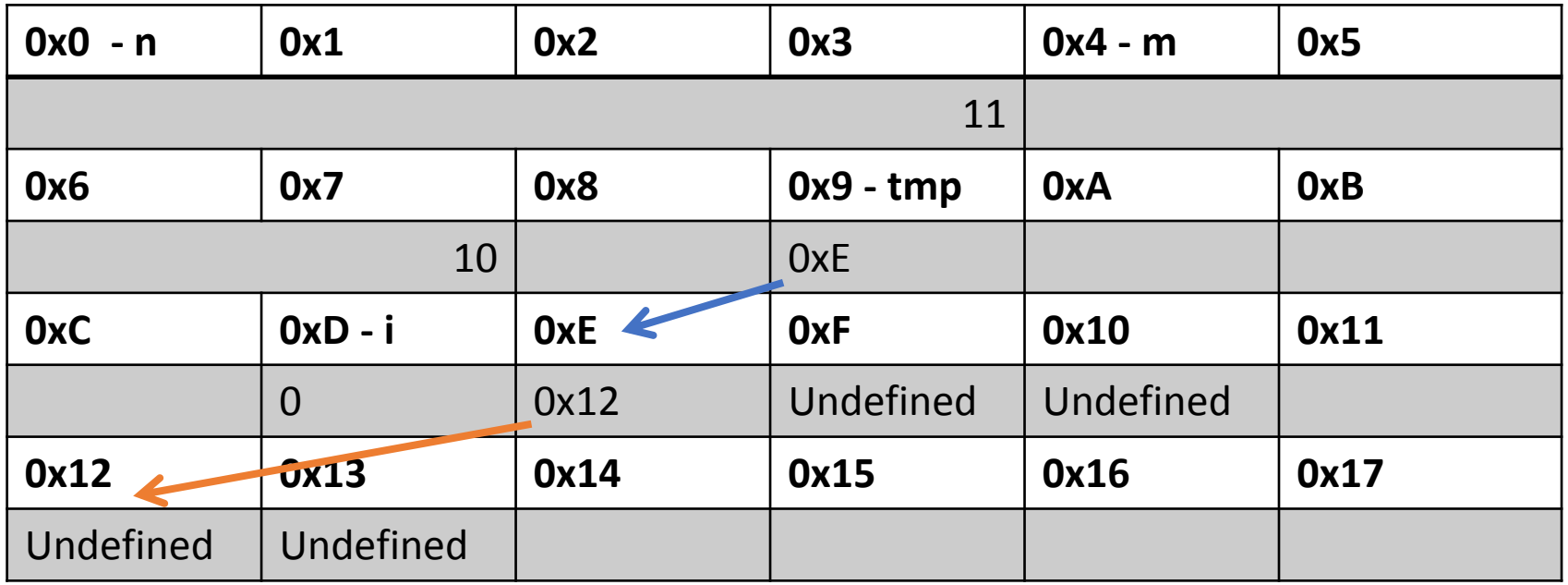
Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```

0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	0	Undefined	Undefined	Undefined	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**

Обработка на динамични данни - визуализация

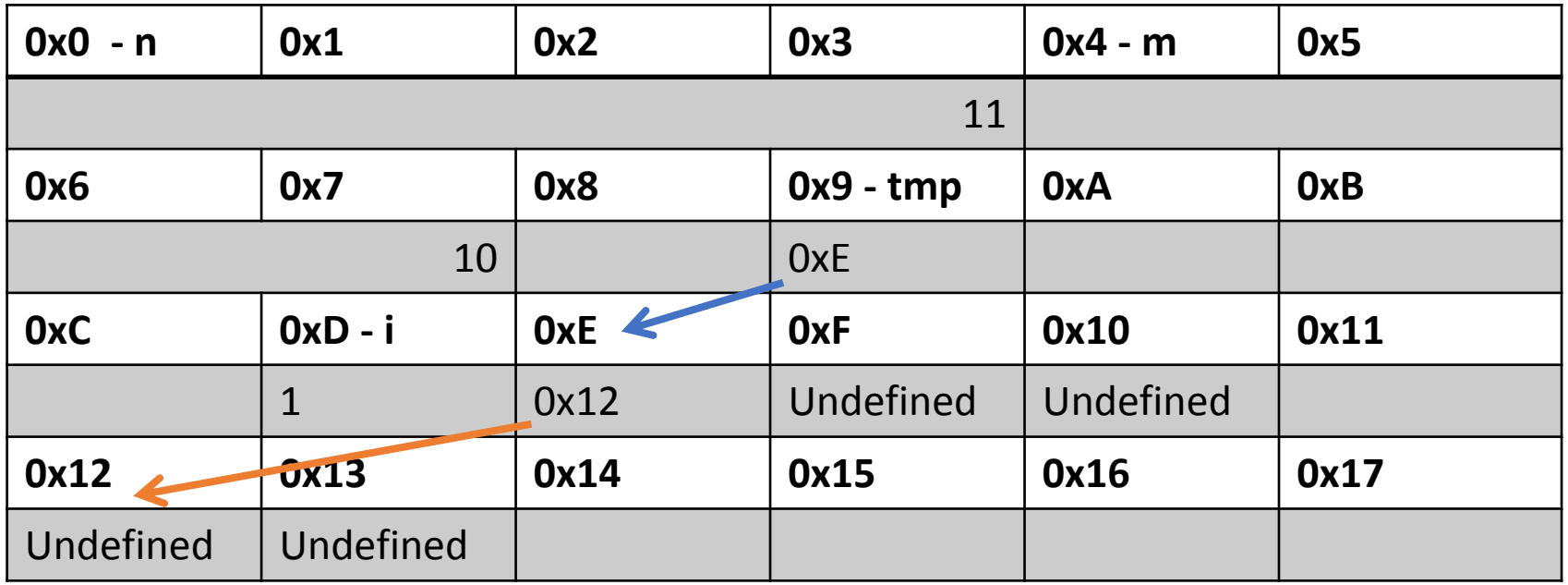
```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```



0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	0	0x12	Undefined	Undefined	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined				

Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```



0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	1	0x12	Undefined	Undefined	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined				

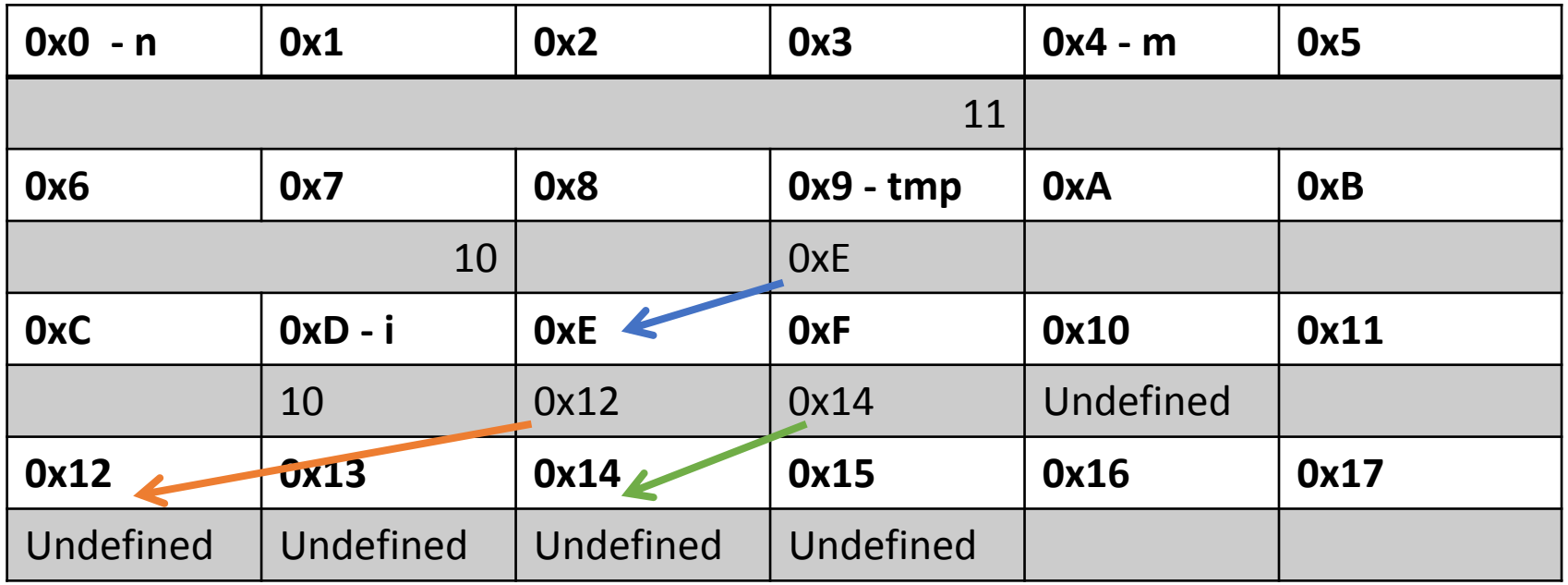
Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```

0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	1	0x12	0x14	Undefined	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined		

Обработка на динамични данни - визуализация

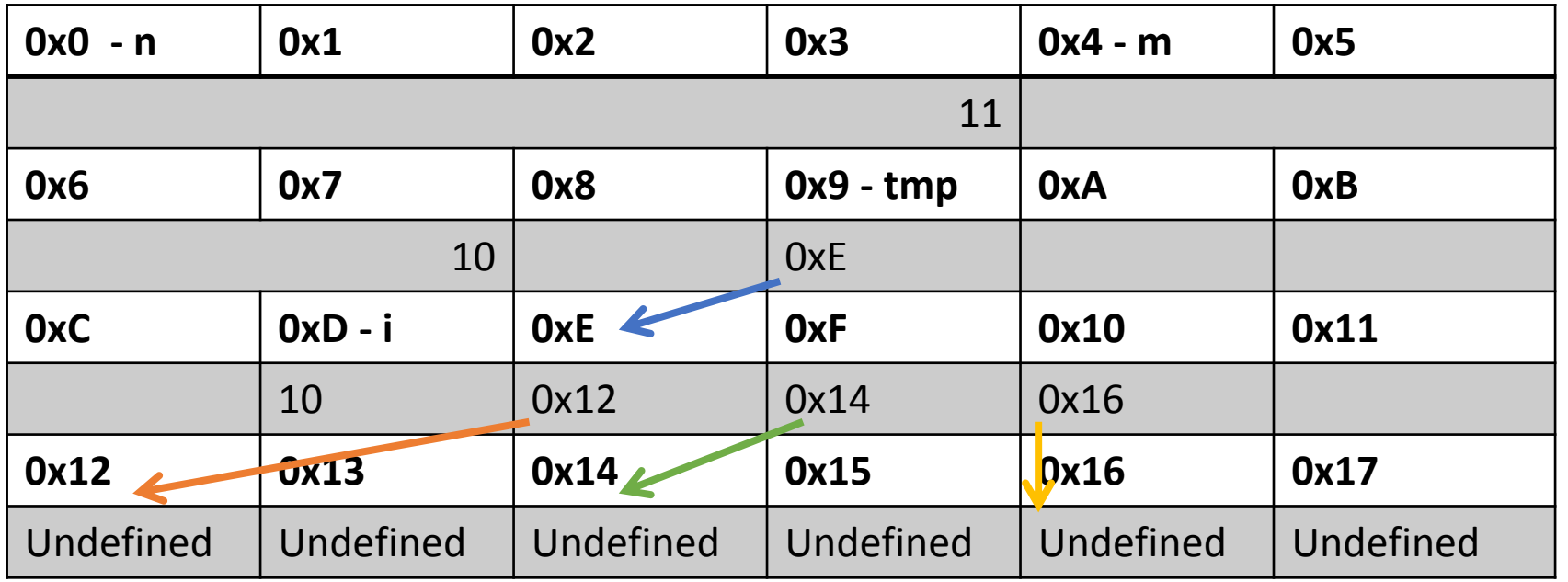
```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```



0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	10	0x12	0x14	Undefined	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined		

Обработка на динамични данни - визуализация

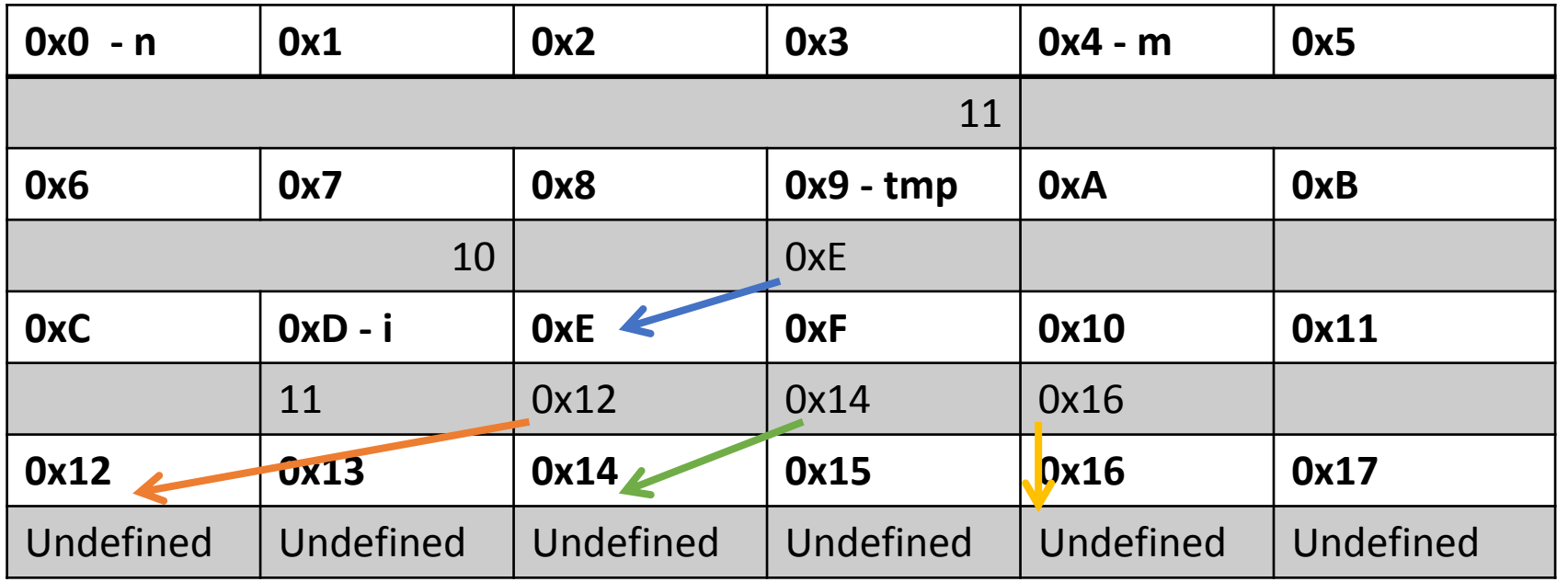
```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```



0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	10	0x12	0x14	0x16	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

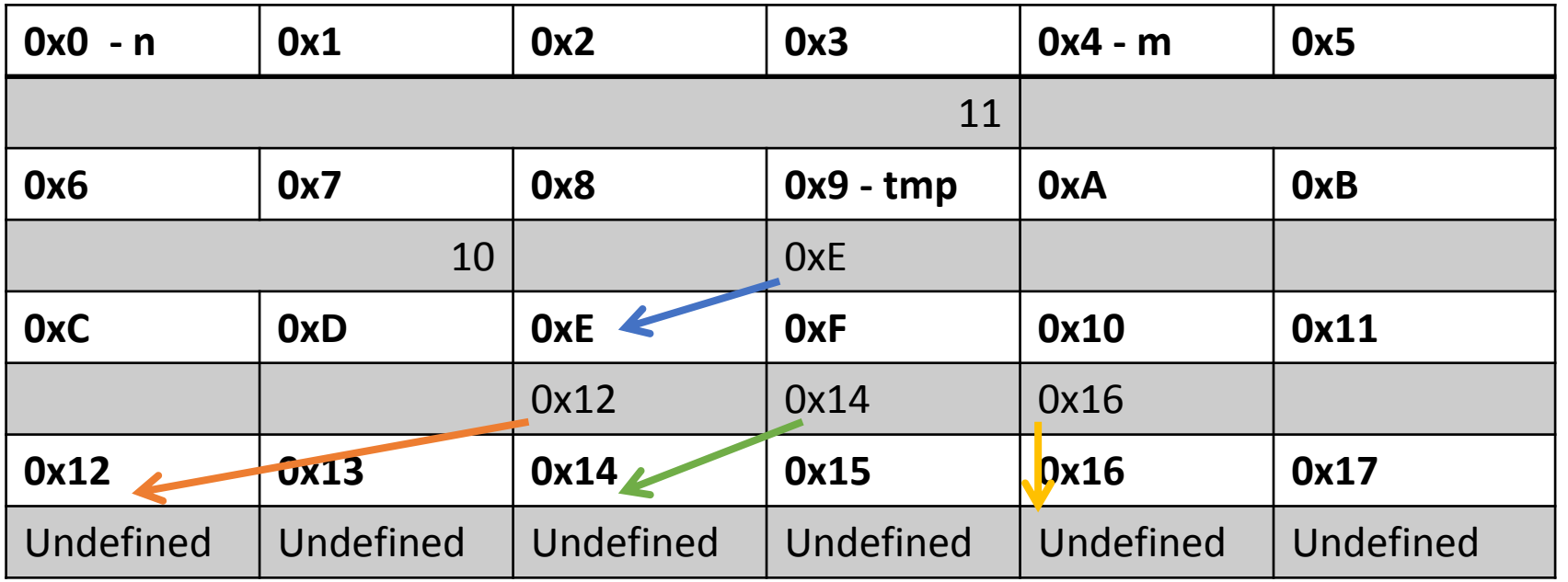
```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```



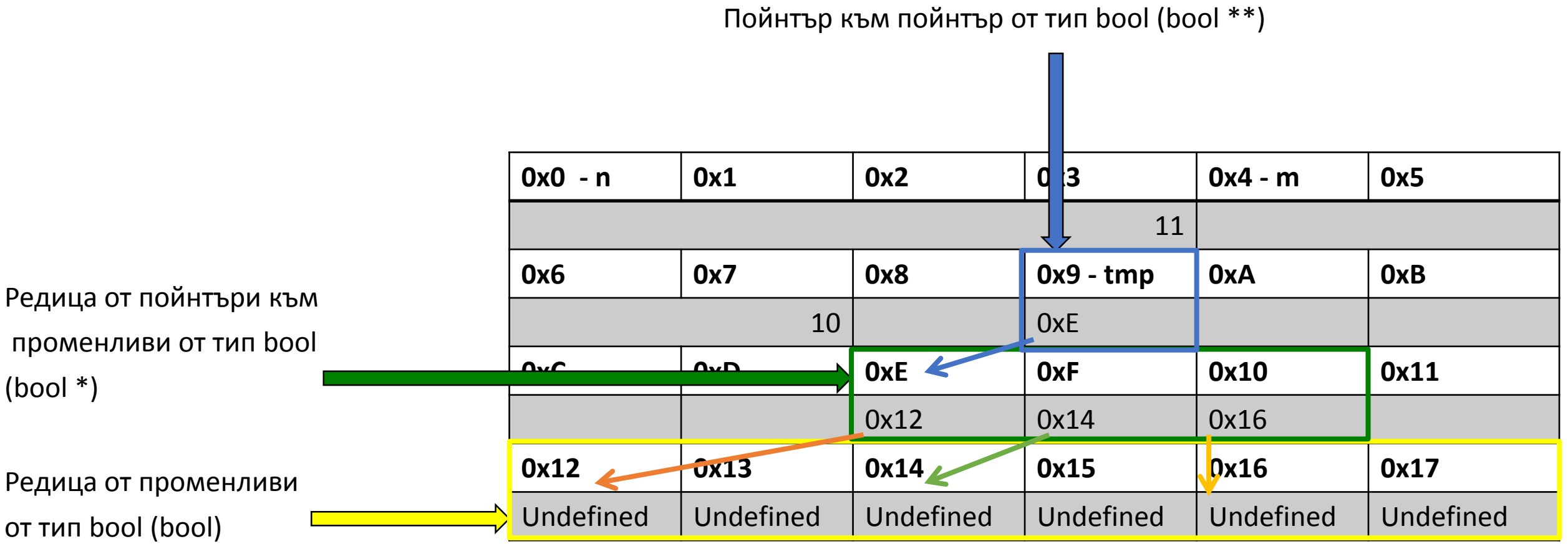
0x0 - n	**0x1**	**0x2**	**0x3**	**0x4 - m**	**0x5**
			11		
0x6	**0x7**	**0x8**	**0x9 - tmp**	**0xA**	**0xB**
	10		0xE		
0xC	**0xD - i**	**0xE**	**0xF**	**0x10**	**0x11**
	11	0x12	0x14	0x16	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```

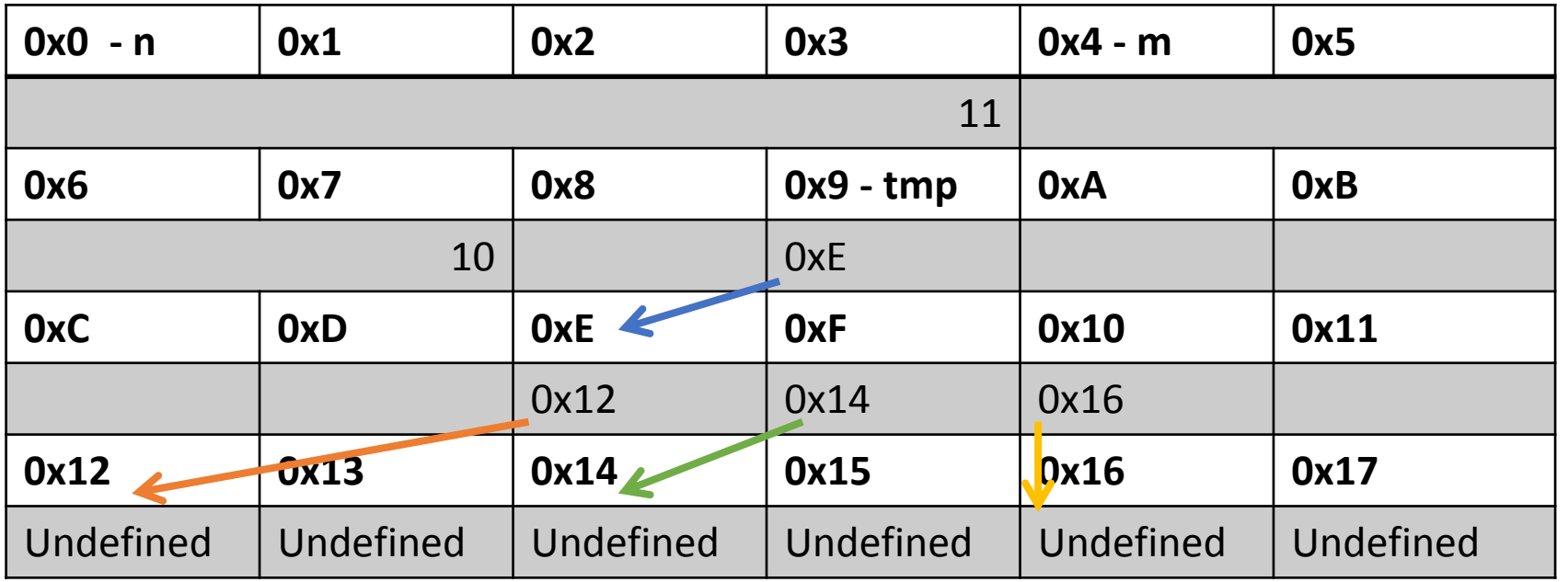


Обработка на динамични данни - визуализация



Обработка на динамични данни - визуализация

```
bool ** Matrix (const unsigned n, const unsigned m)
{
bool ** tmp = new bool * [n];
for(char i = 0; i<n; ++i)
    tmp[i] = new bool [m];
return tmp;
}
```



Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	**0x1**	**0x2**	**0x3**	**0x4**	**0x5**
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB - A**
					0xE
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
		0x12	0x14	0x16	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

Матрица 3X2

0x12	**0x13**
0x14	**0x15**
0x16	**0x17**

0x0	**0x1**	**0x2**	**0x3**	**0x4**	**0x5**
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB - A**
					0xE
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11**
		0x12	0x14	0x16	
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	**0x1**	**0x2**	**0x3**	**0x4**	**0x5**
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB - A**
					0xE
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11 - i**
		0x12	0x14	0x16	0
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	**0x1**	**0x2**	**0x3**	**0x4**	**0x5**
0x6	**0x7**	**0x8**	**0x9**	**0xA**	**0xB - A**
					0xE
0xC	**0xD**	**0xE**	**0xF**	**0x10**	**0x11 - i**
		0x12	0x14	0x16	0
0x12	**0x13**	**0x14**	**0x15**	**0x16**	**0x17**
Undefined	Undefined	Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	0
0x12	0x13	0x14	0x15	0x16	0x17
		Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	1
0x12	0x13	0x14	0x15	0x16	0x17
		Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	1
0x12	0x13	0x14	0x15	0x16	0x17
		Undefined	Undefined	Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	1
0x12	0x13	0x14	0x15	0x16	0x17
				Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	10
0x12	0x13	0x14	0x15	0x16	0x17
				Undefined	Undefined

Обработка на динамични данни - визуализация

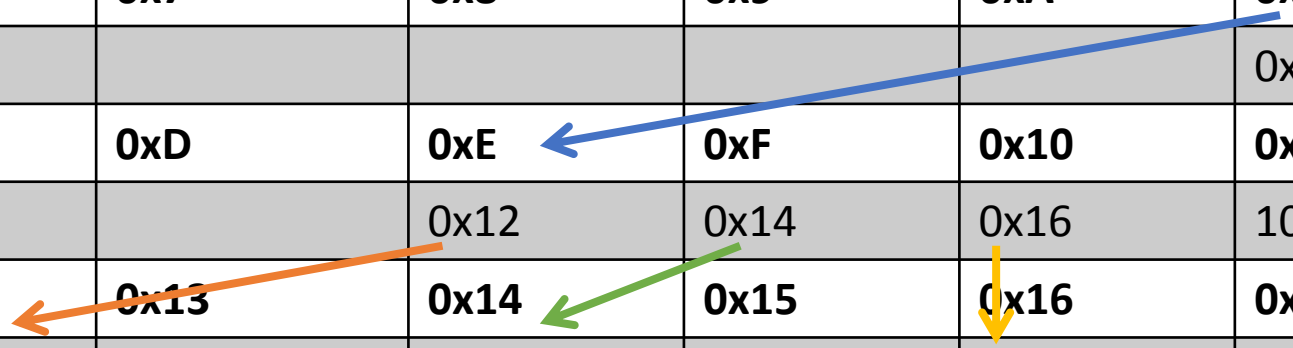
```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	10
0x12	0x13	0x14	0x15	0x16	0x17
				Undefined	Undefined

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

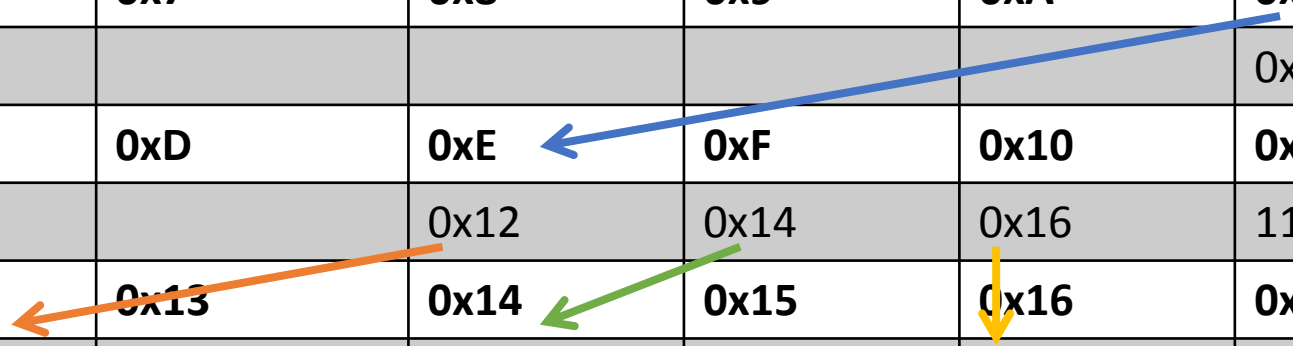
0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	10
0x12	0x13	0x14	0x15	0x16	0x17



Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11 - i
		0x12	0x14	0x16	11
0x12	0x13	0x14	0x15	0x16	0x17



Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```

0x0	0x1	0x2	0x3	0x4	0x5
0x6	0x7	0x8	0x9	0xA	0xB - A
					0xE
0xC	0xD	0xE	0xF	0x10	0x11
		0x12	0x14	0x16	
0x12	0x13	0x14	0x15	0x16	0x17

Обработка на динамични данни - визуализация

```
bool ** A = Matrix(3,2);
for(char i =0; i<3;++i)
{
        delete[] A[i];
}
delete[] A;
```