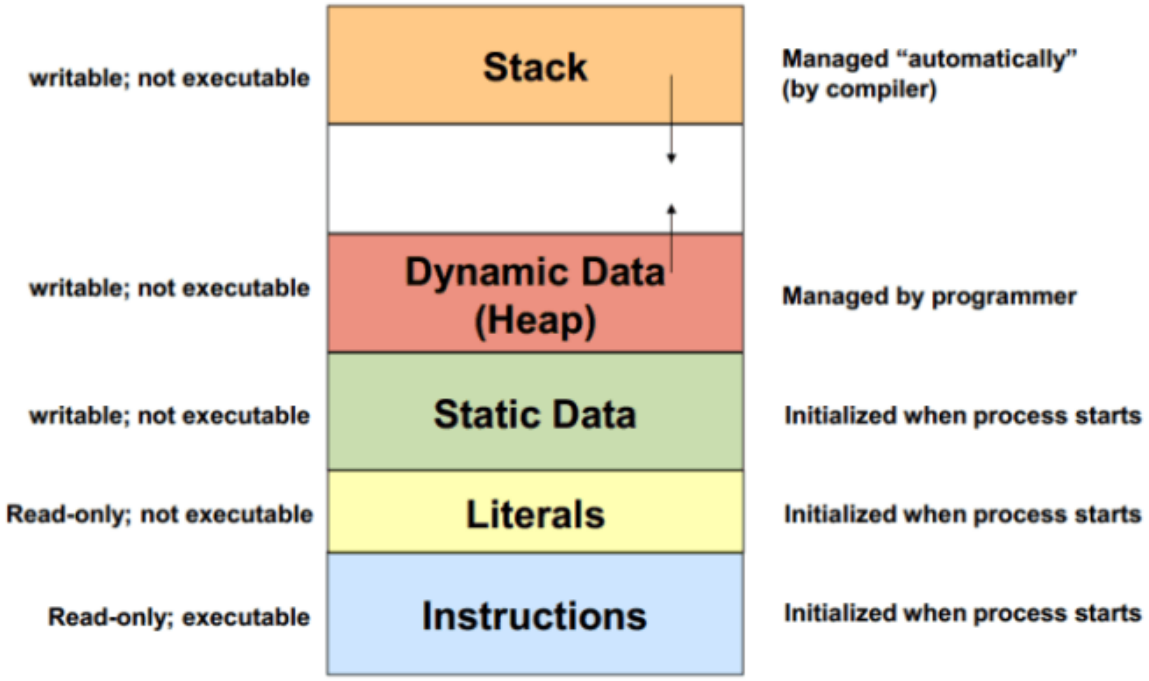
Динамична памет

Нека си припомни различни сегменти на паметта в C++ програми (а и не само)



Когато заделим променлива, тя отива в стековата памет. Тази променлива седи в стека, докато не приключи изпълнението на текущия блок. Припомням, че всеки блок от код, функция, цикъл и т.н. има отделна памет в стековия сегмент, чийто размер е определен **по време на компилация**.

Понякога се случва, да искаме да заделим памет, чийто размер не знаем по време на компилация (пример - ако искаме да въведем оценки - те може да са 2, 3 или 10).

В такива случаи, паметта, която ще заделяме, не се заделя в стековия сегмент, а в **динамичната памет**.

Главната особеност на динамичната памет е, че тя се заделя и освобождава от програмиста, по време на изпълнение.

Програмистът може да прави заявки колко памет иска да задели, но е негова отговорност тази памет да бъде освободена, след като работата с нея е приключена.

Заделяне и освобождаване на динамична памет

Заделянето на динамична памет става по следният начин:

`указател_от_типа* име_на_указателя = new тип_на_данните` - за една променлива от `тип_на_данните`

`указател_от_типа* име_на_указателя = new тип_на_данните(стойност_по_подразбиране)` - за една променлива, със стойност по подразбиране

`указател_от_типа* име_на_указателя = new тип_на_данните[размер_на_масив]` - за масив с даден размер от `тип_на_данните`

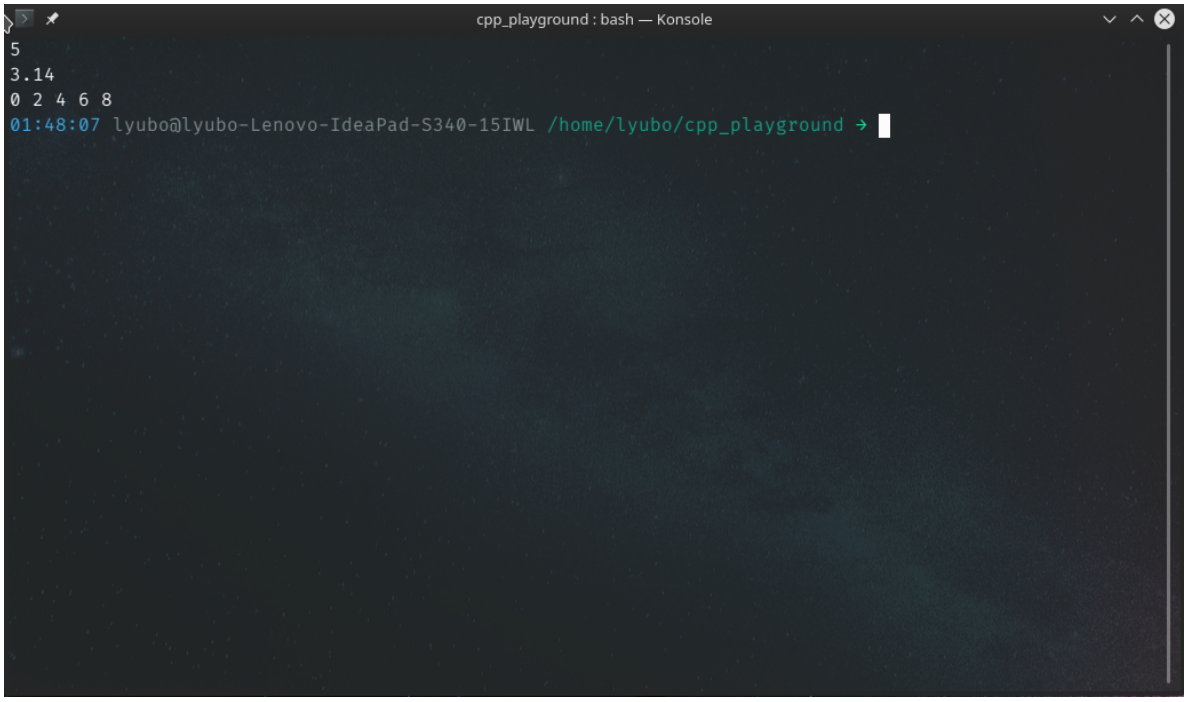
Освобождаването става по следния начин:

`delete име_на_указателя` - за освобождаване на една променлива

`delete[] име_на_указателя` - за освобождаване на масив

Пример:

```
#include <iostream>

using namespace std;

int main()
{
    int* some_number = new int;
    *some_number = 5;

    cout << *some_number << endl;

    delete some_number;


    double* pi = new double(3.14);
    cout << *pi << endl;

    delete pi;

    int* my_array = new int[5];
    for(int i = 0; i < 5; i++)
    {
        my_array[i] = i*2;
    }

    for(int i = 0; i < 5; i++)
    {
        cout << my_array[i] << " ";
    }
    cout << endl;

    delete[] my_array;

    return 0;
}
```

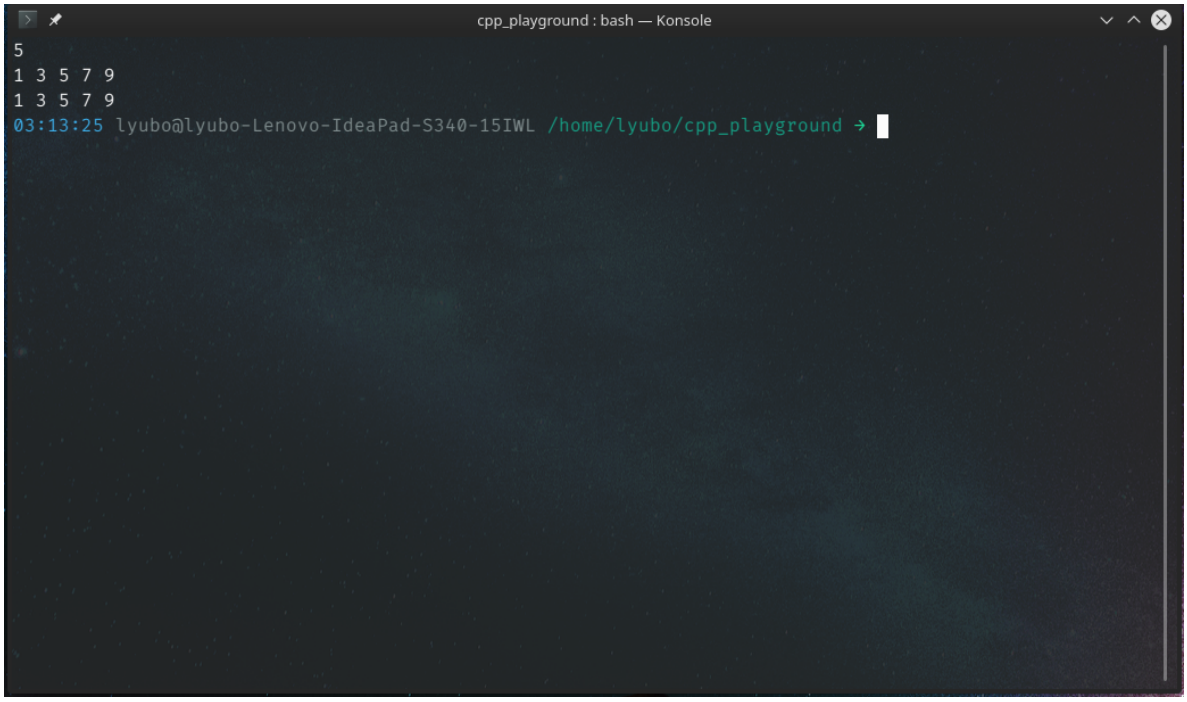


Заделяне на масиви с дължина, въведена по време на изпълнение

Динамичната памет ни позволява да заделим масив с дължина, която не е ясна по време на компилация.

Това става по следният начин:

```
#include <iostream>

using namespace std;

int main()
{
    int array_size = 0;
    cin >> array_size;

    int* array = new int[array_size];

    for(int i = 0; i < array_size; i++)
    {
        cin >> array[i];
    }

    for(int i = 0; i < array_size; i++)
    {
        cout << array[i] << " ";
    }
    cout << endl;

    return 0;
}
```



Задачи

1. Напишете функция, в която потребителя въвежда число `n`, след което въвежда `n`-на брой естествени числа. Да се върне второто най-голямо число и второто най-малко число.
2. Напишете функция, в която потребителя въвежда число от 1 до 12, отговарящо за месец от годината. След това въвежда температурни стойности за всеки един от дните за тови месец (ако месеца = 1, т.е януари, се въвеждат 31, ако е 11, т.е. ноември, се въвеждат 30 стойност). След това потребителя може да въвежда ден от този месец (число от 1 до броя на дните в месеца) като му се показва температурата за този ден, средната температура в интервал от дни [`ден-3`, `ден+3`], разликата между този ден и най-топлия, най-студения и разликата със средната температура за месеца.
3. Напишете функция, която приемаа естествено число `n` - борй на елементите в подаръча кутия. След това потребителя въвежда `n` числа - цени на предметите в кутията. Да се изчисли цената на кутията, ако се дадени следните отстъпки - за предмети на стойност по-голяма от 10 лева: 10% отстъпка, при стойност по-голяма от 20 лева - 15% отстъпка и при стойност по-голяма от 50 лева - 20% отстъпка.
4. Да се напише функция, която изчислява разхода на мастило на принтер - потребителя въвежда естествено число `n`, отговарящо за броя на символите, които трябва да се принтират, число `m` което е броя "принтиращи единици" - п.е. (количество мастило), което е в принтера. След това се въвеждат `n` на брой символа от клавиатурата - в таблицата по-долу е показано всеки символ колко п.е. консумира. Да се изведе необходимото количество п.е. необходимо за принтиране на документа, ако то е налично в принтера, и съобщение, ако мастилото няма да е достатъчно.

Буква	Необходимо п.е.
`!`, `"`, `+`, `-`, `*`, `/`	2
`0`-`9`	3
`;`, `:`, `.`, `=`, `A`-`S`	4
`T`-`Z`	5
`a`-`s`	6
`s-z`	4
всички други символи	3

Пример:

За думата `hello` необходимото п.е. е 30 (6+6+6+6+6)

За текста `AX!~` е необходимо 14 (4+5+2+3)