# Introdução à Programação 2020/2021

António J. R. Neves
Daniel Corujo

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

an@ua.pt / dcorujo@ua.pt

http://elearning.ua.pt/

# Topics





deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

- Dynamic Memory
- new operator
- Standard Template Library

- When we are writing a program, we often don't know how much data it will have to process.
- In these cases, we should only allocate memory as we actually need it at runtime, and release it again as soon as possible.
- This is the principle of dynamic memory management, which also has the advantage that a program doesn't need to be rewritten in order to process larger amounts of data.
- Under normal conditions, it is not possible to allocate more than 8MiB in the stack. For example, code like the following inside a function will give rise to an abnormal program termination:

```
int n = 10000000; // ten million
char s[n];        // too large
```

In these cases one needs to use dynamic memory allocation.

- A good understanding of how dynamic memory really works in C++ is essential to becoming a good C++ programmer.
- Memory in your C++ program is divided into two parts:
  - **stack** − All variables declared inside the function will take up memory from the stack.
  - **heap** − This is unused memory of the program and can be used to allocate the memory dynamically when program runs.
- We can allocate memory at run time within the heap for the variable of a given type using the special operator **`new`** which returns the address of the space allocated.
- If you are not in need of dynamically allocated memory anymore, you can use **`delete`** operator.

```
// Example 1
char* pvalue  = NULL;          // Pointer initialized with null
pvalue  = new char[20];        // Request memory for the variable
...
delete [] pvalue;              // Delete array pointed by pvalue

//Example 2
int** a = new int*[3];         // Allocate memory for a 3x4 array
for(int i = 0; i < 3; ++i)
    a[i] = new int[4];
...
for(int i = 0; i < 3; ++i) { // Delete array pointed by pvalue
    delete [] a[i];
}
delete [] a;

// Generally
ObjectType *obj = new ObjectType;
```

- The size of the executable file of the following program

```
int a[1000000];
int main(void)
{
  return a[3];
}
```

  is very small (less than 10KiB)

- The size of the executable file of the following program

```
int a[1000000] = { 3 };
int main(void)
{
  return a[3];
}
```

  is very large (4MiB).

- So, if possible, do not initialize large global variables.

| | | |
|---|---|---|
| highest addr | kernel memory area (inaccessible) | not used |
| | stack (grows to lower addresses) | for local variables and other function data |
| | empty | shrinks when the stack or heap grows |
| | heap (grows to higher addresses) | for dynamic memory allocation |
| | persistent data (read-write) | for global variables |
| | initialized persistent data (read only) | for constants |
| | empty | not used |
| | text (code) | program code |
| lowest addr | empty | guard region (never used) |

- Talk about dynamic memory lead us to think about the STL. Why?
- Most computer programs exist to process data that may represent a wide variety of real-world information
- Whatever it represents, data is stored in memory and manipulated in similar ways
- University computer science programs typically include a course called “Data Structures and Algorithms”
- The term data structures refers to the ways data is stored in memory, and algorithms refers to how it is manipulated
- Standard C++ includes its own built- in container class library called the Standard Template Library (STL)
- STL can be used as a standard approach to storing and processing data
- Introduction to Standard Template Library in C++

- The three most important entity in STL are containers, algorithms, and iterators
  - A container is a way that stored data is organized in memory - the STL containers are implemented by template classes, so they can be easily customized to hold different kinds of data
  - Algorithms in the STL are procedures that are applied to containers to process their data in various ways (example: sort, copy, search, and merge data) - algorithms are represented by template functions
  - Iterators are a generalization of the concept of pointers: they point to elements in a container - are a key part of the STL because they connect algorithms with containers

- A container is a way to store data, whether the data consists of built-in types such as int and float, or of class objects
- The STL makes seven basic kinds of containers available, as well as three more that are derived from the basic kinds
- We can create our own container based on the basic kinds
- Two main categories:
  - sequence - vector, list, and deque (and stack, queue, and priority queue)
  - associative - set, multiset, map, and multimap
- Why not use C++ arrays in all data storage situations? The answer is efficiency and code simplicity…