Utilizzo della memoria heap in C++

Liceo G.B. Brocchi – Bassano del Grappa (VI)
Liceo Scientifico – opzione scienze applicate
Giovanni Mazzocchin

`malloc`, `calloc`, `realloc` e `free`

- Le funzioni della libreria standard C `malloc`, `calloc`, `realloc` e `free` permettono di gestire la memoria heap
- Avete sicuramente notato che la loro interfaccia non è particolarmente comoda per il programmatore
- Il C++ mette a disposizione 2 operatori integrati nel linguaggio che permettono di fare sostanzialmente le stesse cose, ma con un interfaccia più semplice:
  - **`new`**
  - **`delete`**
- `new` e `delete` sono operatori integrati nel linguaggio, quindi non serve includere niente di particolare

L'operatore new

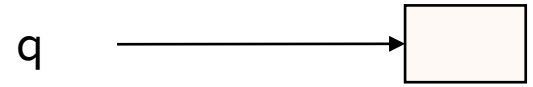
```
int *q = new int; //allocates 1 int on the free store/heap
```

```
double *alloc_double_array(unsigned int n, int val) {
  double *array = new double[n];
  //allocated an array of n doubles on the free store
  for (int i = 0; i < n; i++) {
    array[i] = val;
  }
  return array;
}
```

sotto il tappeto ci sono `malloc` e `calloc`. Ma l'interfaccia è più chiara e tipizzata. All'operatore `new` si può chiedere direttamente di allocare un certo numero di elementi del tipo desiderato

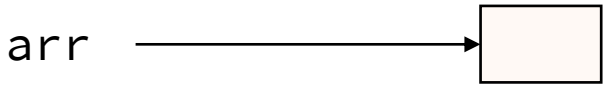
L'operatore new

```
int *q = new int; //memory could be uninitialized
```



`sizeof(int)` celle sull'heap

```
double *arr = new double[4];
```



`4 * sizeof(double)` celle sull'heap

Attenti ai tipi

```
type_errs.cpp: In function ‘int main()’:
type_errs.cpp:11:31: error: cannot convert
‘double*’ to ‘char*’ in initialization
   11 |    char *p1 = new double(3.1415);
      |
```

```
#include <iostream>
#include <cstdio>
using namespace std;

int main() {
  //correct typing
  double *p = new double(3.1415);
  printf("%6.4f\n", *p);

  //wrong typing
  char *p1 = new double(3.1415);
}
```

Attenti ai tipi

con un *typecast* sto dicendo al compilatore: voglio assolutamente fare questa conversione, non disturbarmi

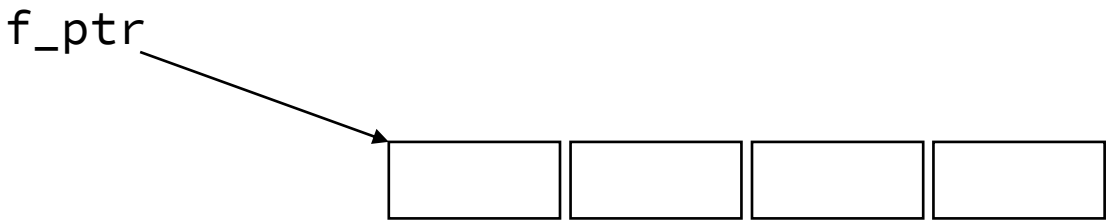
```
//right typing
float *f_ptr = new float(3.1415);

//forcing the conversion with a typecast, C
void *v_ptr1 = (void*) f_ptr;

//forcing the conversion with a typecast, C++
void *v_ptr2 = static_cast<void*>(f_ptr);
```

Attenti ai tipi

grazie al casting da `float*` a `void*`, posso accedere ai singoli byte utilizzati per rappresentare 3.1415 sulla macchina. Sul mio computer, `sizeof(float)` è 4 byte



L'operatore `delete`

```
delete p; /* frees the memory for the object pointed to by p,
             allocated with the new operator */

delete[] p1; /* frees the memory for an array of objects of type T,
                allocated by new T[], pointed to by p1 */
```

Da vedere a casa

- Garbage Collection (Mark & Sweep) - Computerphile