سلسلة تعلم البرمجة بلغة C++ الحديثة

Learn Modern C++ Programming Course

إعداد المهندس أحمد الديب

#7: Pointers

Pointers

```
int a1{10};
int* a2{&a1}; // pointer to a1.
int a3{*a2}; // getting int value a2 points to.

int** a4; // pointer to pointer
int* a5[10]; // pointer to an array with 10 ints.

// pointer to function multiply
int (*f1)(int*, int*){multiply};
```

Source: The C++ Programming Language (4th Edition), Bjarne Stroustrup

void*

- Pointer to an object of unknown type.
- Used when passing an address of a memory location without actually knowing what type of object is stored there.
- Pointer to any type of object can be assigned to a variable of type void*, but a pointer to function or a pointer to member cannot.
- Can be explicitly converted to another type (casted). Can be unsafe because the compiler cannot know what kind of object is really pointed to.

Source: The C++ Programming Language (4th Edition), Bjarne Stroustrup

void* Example

```
int* pi;
void* pv = pi; // ok: implicit conversion of int* to void*

// error: ISO C++ does not allow
// indirection on operand of type 'void *'
*pv;

// error: expression must be a pointer
// to a complete object type
++pv;

int* pi2 = static_cast<int*>(pv); // explicit conversion back to int*

double* pd1 = pv; // error
double* pd2 = pi; // error
double* pd3 = static_cast<double*>(pv); // unsafe
```

Source: The C++ Programming Language (4th Edition), Bjarne Stroustrup

nullptr

- Represents the null pointer, that is, a pointer that does not point to an object.
- Before `nullptr` was introduced, zero (0) was used.
- It has been popular to define a macro NULL to represent the null pointer.
- In C, NULL is typically `(void*)0`, which makes it illegal in C++.

```
int* n1; // Not a null pointer

// using nullptr is more readable
int* n2 = nullptr;
void* n3 = nullptr;

int* n4 = 0; // Old
int* n5 = NULL; // Old, taken from C
```

Source: The C++ Programming Language (4th Edition), Bjarne Stroustrup

Pointers and const

- C++ supports two notions of immutability
  `const`: meaning roughly “I promise not to change this value”.
  `constexpr`: meaning roughly “to be evaluated at compile time”

```
int x = 10;
const int* p1; // pointer to constant
int* const p2 = &x; // constant pointer
const int* const p3 = &x; // constant pointer to constant

p1 = &x; // ok
*p1 = 20; // error
p2 = &x; // error
*p2 = 10;

const int y = 0;
int* const p3 = &y; // error: object is const
```

Source: The C++ Programming Language (4th Edition), Bjarne Stroustrup

Thank you