C++ University
Session 7

Tristan Brindle

Feedback and Communication

- Your feedback is vital
- Otherwise, we don’t know what you don’t know!
- Please join the #cpplondonuni channel on the cpplang Slack — Go to https://cpplang.now.sh/ for an “invitation”

Next week: lightning talks!

- Next week, we’d like *you* to run the session!
- Pick a C++-related topic you don’t feel comfortable about, and prepare a short (5 minute) talk on the subject
- Don’t worry about picking something “too basic” (or too advanced!) — we want to encourage people of all experience levels to contribute 🙂

Today’s Lesson Plan

- Noughts and Crosses exercise code reviews
- Pointers 101

Noughts and Crosses

- Last week's group exercise was to write a two-player noughts and crosses game

  https://github.com/CPPLondonUni/noughts_and_crosses

- "Homework" was to finish off the exercise
- Any volunteers to show us their code?

Pointers 101

“Oh no, pointers! 😩”

–Programmer before learning C++

“Oh, no pointers! 😃”

–Programmer after learning C++

Pointers 101

- In C++ a pointer is a variable represents the *memory address* of some other variable (or function)
- These are often called “raw pointers”, to differentiate them from “smart pointers”, which we’ll talk about later
- In modern C++ there are three main uses for raw pointers:
    - As a kind of nullable reference — that is, a reference which may not point to anything
    - As an iterator for arrays
    - When dealing with raw memory for low-level work
- A good rule of thumb: *use references when you can, pointers when you have to*

Pointers 101

- We can take the memory address of a variable using an ampersand (&) before the variable name, for example

```
int i = 0;
auto p = &i; // p contains the memory address
             // of the variable i
```

- The standard library function `std::addressof(x)` can be used for the same thing, and avoids problems with overloaded `operator&`.

Pointers 101

- *Pointers aren't magical!*
- A pointer behaves in many ways just like an unsigned integer type which is large enough to contain a memory address
- This means that we can, for example, take the address of a pointer:

```
int i = 0;
auto p = &i; // p contains the memory address of i
auto pp = &p; // pp contains the memory address of p
```

Pointers 101

- We can declare a variable of pointer type by putting an asterisk after the type name, for example

```
int i = 0;
int* p; // declares a pointer to an int, uninitialized
p = &i; // p now contains the address of i
```

- In this example, we say that the variable p has type “pointer to int”

Pointers 101

- Unlike references, pointers can be changed to point to a different memory address once initialised. For example

```
int i = 0;
int j = 1;
int* p = &i; // p contains the address of i
p = &j; // p now contains the address of j
```

Null pointers

- Any pointer type can be assigned the special value `nullptr`, meaning “does not point to anything”. For example:

```
int* p = nullptr; // p is a null pointer to int
```

- In C and older C++ code, the macro `NULL` is used for this purpose

- Remember, *always initialize your variables*! If there is no valid value for this pointer (yet), then use `nullptr`

Dereferencing pointers

- We can *dereference* a pointer to obtain the value of the variable it points to

- This is done by saying *p, where p is a pointer. For example:

```c
int i = 0;
int *p = &i;
*p = 4;
// i now has the value 4
```

- **Never, ever dereference an invalid pointer!**

Exercises

- https://github.com/CPPLondonUni/pointers101

“Homework”

- Lightning talks!

Online Resources

- https://isocpp.org/get-started
- cppreference.com — The bible, but aimed at experts
- cplusplus.com — Another reference site, also has a tutorial section
- learncpp.com — Free online tutorial, very up-to-date
- https://www.pluralsight.com/authors/kate-gregory - Comprehensive set of courses from an experienced C++ trainer (free trial)
- reddit.com/r/cpp_questions
- Cpplang Slack channel — https://cpplang.now.sh/ for an “invite”
- StackOverflow (but…)

Thanks for coming!

C++ London University:

- Website: cpplondonuni.com
- Github: github.com/CPPLondonUni

Where to find Tom Breza:

- On Slack: cpplang.slack.com #learn #cpplondon
- E-mail: tom@PCServiceGroup.co.uk
- Mobile: 07947451167

My stuff:

- Website: tristanbrindle.com
- Twitter: @tristanbrindle
- Github: github.com/tcbrindle

See you next time! 🙂