Università degli Studi di Padova

Pointers, references, callbacks

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB

Agenda

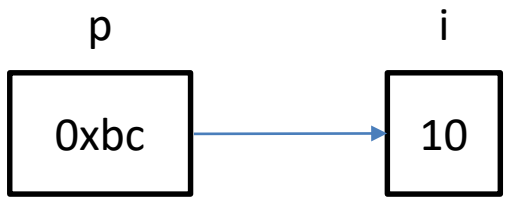
- Pointers
- References
- Cast
- Callbacks

Pointers

- Variables that store memory addresses
- C/C++: pointers to a type
  - Needed to properly deal with the memory at destination
  - Provide direct access to the memory location
    - Write access!

```
int i = 10;

int *p = &i;
```



Memory address of a variable

Working with pointers

- Pointers need to be dereferenced to access their content
  - How to write to the destination pointed by p?
  - Dereference operator (*)

```
int i = 10;

int *p;

p = &i;

*p = 15;
```

```
int i = 10;

int *p = &i;

*p = 15;
```

References

- References are an alternative method
  - Create aliases
  - Different syntax, similar concept

```
int i = 10;

int &r = i;

r = 15;
```

Functions with pointers and references

Int

```
void f(int i);

int main(void)
{
  int i = 0;

  f(i);

  return 0;
}

void f(int i)
{
  i += 2;
}
```

Pointer

```
void f(int *p);

int main(void)
{
  int i = 0;

  f(&i);

  return 0;
}

void f(int *p)
{
  *p += 2;
}
```

Reference

```
void f(int &r);

int main(void)
{
  int i = 0;

  f(i);

  return 0;
}

void f(int &r)
{
  r += 2;
}
```

Functions with pointers and references

Int

```
void f(int i);

int main(void)
{
  int i = 0;

  f(i);

  return 0;
}

void f(int i)
{
  i += 2;
}
```

Pointer

```
void f(int *p);

int main(void)
{
  int i = 0;

  f(&i);

  return 0;
}

void f(int *p)
{
  *p += 2;
}
```

Reference

```
void f(int &r);

int main(void)
{
  int i = 0;

  f(i);

  return 0;
}

void f(int &r)
{
  r += 2;
}
```

Casts

- A cast is a method for changing the type of a variable

C version

```
int i = 10;

float f;

f = (float) i;
```

C++ version

```
int i = 10;

float f;

f = static_cast<float> (i);
```

Cast to void*

- Void* is a generic pointer
- Used to pass a generic object
  - C-style generic argument
- Needs a double cast
  - Object* -> void* and void* -> object*
- This might be requeste by OpenCV

Callbacks

- "A callback is any executable code that is passed as an argument to other code that is expected to call back (execute) the argument at a given time" (Wikipedia)
- Used to handle events in GUIs
  - E.g.: mouse events (like click)
- OpenCV GUI can handle callbacks

Mouse callback

```
#include <iostream>
// use this directly to include all modules instead of the single headers
//#include <opencv2/opencv.hpp>
#include <opencv2/core.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>

#define NEIGHBORHOOD_Y 9
#define NEIGHBORHOOD_X 9
#define MAX_B_CHANNEL 70
#define MAX_G_CHANNEL 100
#define MAX_R_CHANNEL 70


int main(int argc, char** argv)
{
  cv::Mat input_img = cv::imread("../data/robocup.jpg");

  cv::resize(input_img, input_img, cv::Size(input_img.cols / 2.0, input_img.rows / 2.0));
  cv::imshow("img", input_img);
  cv::setMouseCallback("img", onMouse, (void*)&input_img);

  cv::waitKey(0);
  return 0;
}
```

Mouse callback

```
void onMouse( int event, int x, int y, int f, void* userdata){

  // If the left button is pressed
  if (event == cv::EVENT_LBUTTONDOWN)
  {
    // Retrieving the image from the main
    cv::Mat image = *(cv::Mat*) userdata;
    cv::Mat image_out = image.clone();

    // Preventing segfaults for looking over the image boundaries
    if (y + NEIGHBORHOOD_Y > image_out.rows
        || x + NEIGHBORHOOD_X > image_out.cols)
      return;

    // Mean on the neighborhood
    cv::Rect rect(x, y, NEIGHBORHOOD_X, NEIGHBORHOOD_Y);
    cv::Scalar mean = cv::mean(image_out(rect));
    std::cout << "Mean: " << mean << std::endl;
```

Università degli Studi di Padova

Pointers, references, callbacks

Stefano Ghidoni

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

INTELLIGENT AUTONOMOUS SYSTEMS LAB