COMP26020 Programming Languages and Paradigms Part 1: C Programming

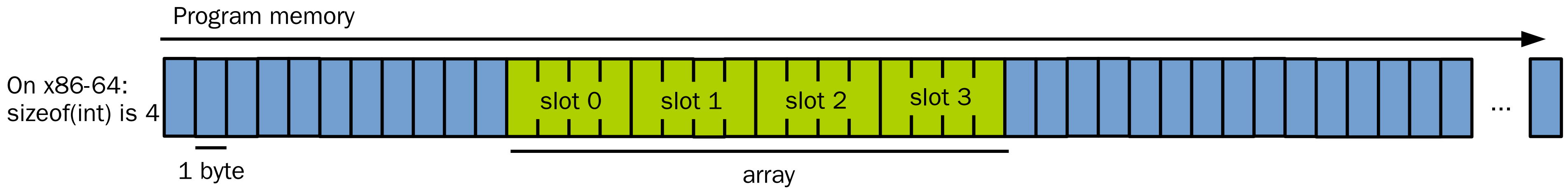
Arrays, Strings, Command Line Arguments

Arrays

```
int array[4];      // declaration

array[0] = 10;     // write a slot
array[1] = 20;
array[2] = 30;
array[3] = 40;

int x = array[3];   // read a slot
```

```
printf("array[%d] contains %d\n", 0, array[0]);
printf("array[%d] contains %d\n", 1, array[1]);
printf("array[%d] contains %d\n", 2, array[2]);
printf("array[%d] contains %d\n", 3, array[3]);



//                 05-c-arrays-strings-cmdline/array.c
```

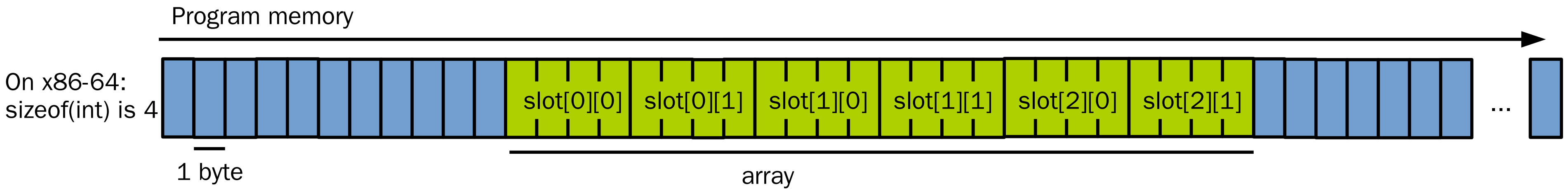
Arrays are stored contiguously in memory:



Multi-Dimensional Arrays

```c
int my_2d_array[3][2];

my_2d_array[0][0] = 0;
my_2d_array[0][1] = 1;
my_2d_array[1][0] = 2;
my_2d_array[1][1] = 3;
my_2d_array[2][0] = 4;
my_2d_array[2][1] = 5;

printf("my_2d_array[%d][%d] = %d\n", 0, 0, my_2d_array[0][0]);
printf("my_2d_array[%d][%d] = %d\n", 0, 1, my_2d_array[0][1]);
printf("my_2d_array[%d][%d] = %d\n", 1, 0, my_2d_array[1][0]);
printf("my_2d_array[%d][%d] = %d\n", 1, 1, my_2d_array[1][1]);
printf("my_2d_array[%d][%d] = %d\n", 2, 0, my_2d_array[2][0]);
printf("my_2d_array[%d][%d] = %d\n", 2, 1, my_2d_array[2][1]);
```

05-c-arrays-strings-cmdline/array-2d.c



Arrays

- Static initialisation allows to omit the size (of the first dimension)

```c
int array[] = {1, 2, 3, 4, 5};
int array_2d[][2] = { {1, 2}, {3, 4}, {5, 6} };

printf("array[4] = %d\n", array[4]);
printf("array_2d[2][0] = %d\n", array_2d[2][0]);
```

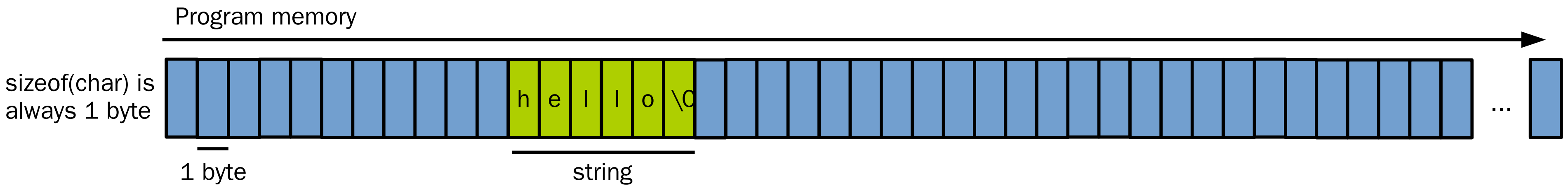
05-c-arrays-strings-cmdline/array-init-static.c

Character arrays, i.e. Strings

- String in C: **array of characters** that ends with the special `\0` end of string character marker

```c
char string1[6] = "hello";
char string2[] = "hello";
char string3[6];

string3[0] = 'h';
string3[1] = 'e';
string3[2] = 'l';
string3[3] = 'l';
string3[4] = 'o';
string3[5] = '\0'; // Important here
```

```c
printf("%s\n", string1);
printf("%s\n", string2);
printf("%s\n", string3);
//        05-c-arrays-string-cmdline/array-init-static.c
```



Command Line Arguments

$$\underbrace{\texttt{gcc}}_{\text{program}}\ \underbrace{\texttt{test.c -o program}}_{\text{arguments}}$$

Command Line Arguments

- `main` parameters:
  - `argc`
  - `argv`: array containing the command line arguments

```c
int main(int argc, char **argv) { // 'char ** argv' means 'char argv[][]'

    printf("Number of command line arguments: %d\n", argc);

    // This is a bit dangerous...
    printf("argument 0: %s\n", argv[0]);
    printf("argument 1: %s\n", argv[1]);
    printf("argument 2: %s\n", argv[2]);

    return 0;
}
```

05-c-arrays-string-cmdline/cmdline-args.c

Command Line Arguments

- Arguments are strings, to convert to integer use `atoi`:

```c
#include <stdio.h>
#include <stdlib.h> // needed for atoi

/* Sum the two integers passed as command line integers */
int main(int argc, char **argv) {
    int a, b;

    // Once again, dangerous!
    // We'll fix that in the next lecture regarding control flow
    a = atoi(argv[1]);
    b = atoi(argv[2]);

    printf("%d + %d = %d\n", a, b, a+b);

    return 0;
}
```

05-c-arrays-string-cmdline/atoi.c

- To convert a string to a `double` use `atof`

Summary

- Arrays
- Strings
- Command line arguments

Feedback form: https://bit.ly/3xrfCAt