COP 3223H: Introduction to C Programming

Fall 2023

University of Central Florida

Dr. Kevin Moran

Week 1 - Class 3: C Variables & Data Types

Administrivia

- Entrance Survey has been posted to Webcourses!
  - Due ***today*** at 11:59pm (should only take a few mins)
- Please register for Ed Discussions!
- Eustis assignment will be posted today
  - Due next Friday (Sept 1st)
- Syllabus Quiz will be posted today
  - Also Due Next Friday (Sept 1st)

Today's Agenda

- Discuss variables in C
- Discuss some useful C data types
- See these in action

Variables in C

User-defined Identifiers

- We choose our own identifiers to name memory cells that will hold data and program result and to name operations that we define.
- Rules for User-Defined Identifiers
  - An identifier must consist only of letters, digits, and underscores.
  - An identifier cannot begin with a digit.
  - A C reserved word cannot be used as an identifier.
  - An identifier defined in a C standard library should not be redefined.
- We will use CamelCase for structs, and snake_case for variables/ functions.

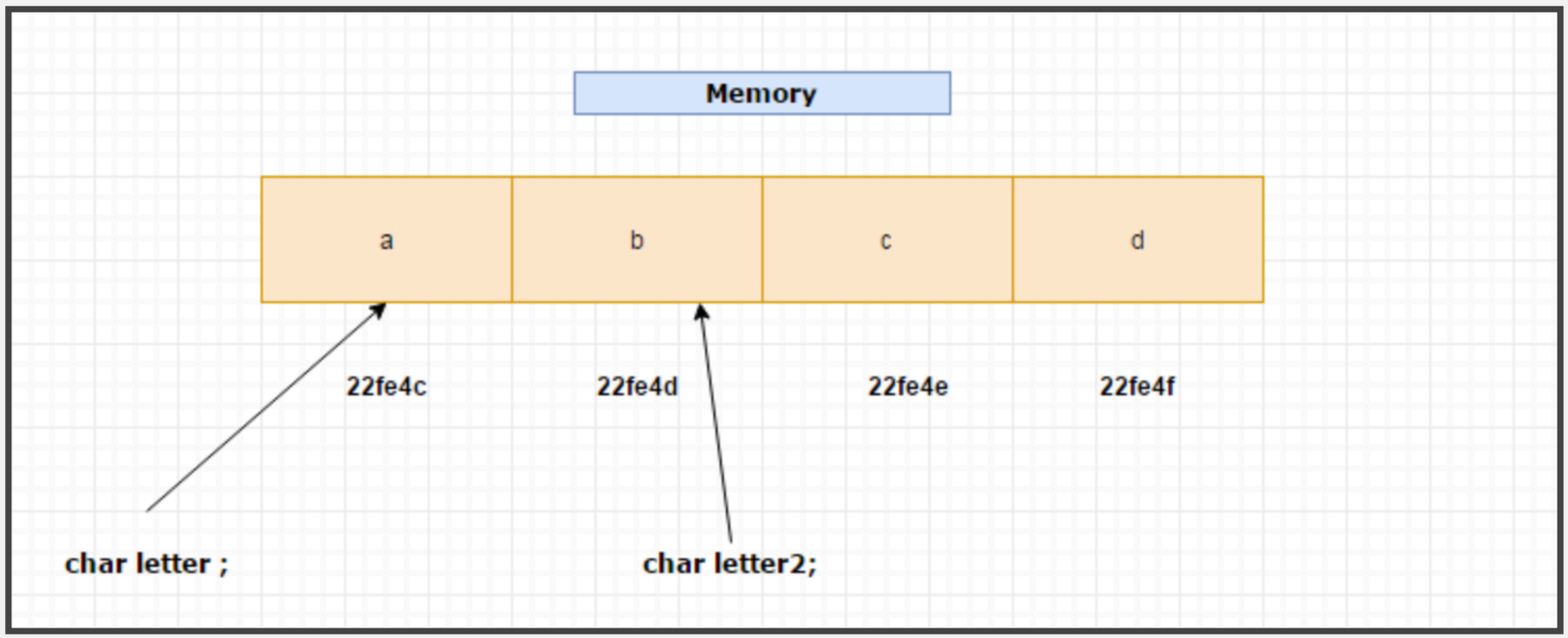
Variables

- Variables are names associated with a memory cell whose value can change.
  - User-Defined Identifiers
- Variable Declarations are statements that communicate to the compiler the names of variables in the program and the kind of information stored in each variable.
  - Syntax
    - int *variable_list;*
    - double *variable_list;*
    - char *variable_list;*

	A	B	C
0	`int x = 0`		
1			`double num = 1.4`
2			
3	`char letter = ‘a’`		
4			

Why Variables

- Because as programmers, it would be difficult for us to keep track of memory addresses 🙂



Sample Syntax for Declaring Variables

```
int val;

double x;

float y;

char letter;
```

Data Types in C

Data Types

- A set of values and operations that can be performed on those values.
  - Types of Data that can be stored in C:
    1. `int` – integer numbers
    2. `double` – decimal numbers
    3. `float` – similar to double BUT different amount of allocation for memory storage (smaller allocation)
    4. `char` - a character from the keyboard

Type	Range in Typical Implementation
`int`	-2,147,483,647 ... 2,147,483,647
`double`	10^{-307}...10^{308} (15 significant digits)
`float`	10^{-37}...10^{38} (6 significant digits)

`double` and `float` Data Types

- Most beginners think that doubles and floats can be used interchangeably.
  - THIS IS FALSE!!!
- doubles have twice the precision of float type values.
- If they are used interchangeably, then you will likely encounter rounding errors.
- *When in doubt, always use double for extra precision!!!!! If any programming problem does not specify the data type for any real number, use double!!!*

`char` Data Type

- Data type char represents an individual character value: letter, digit, or a special symbol
  - Ex: ‘A’, ‘z’, ‘2’, ‘9’, ‘*’, ‘:’, ‘”’, ‘ ’
- Characters are represented uniquely in memory as an integer for the system to properly evaluate.
  - The value is known as ASCII Value
  - This can be utilized when comparing characters.

Character	ASCII Code
’ ‘	32
*’	42
‘A’	65
‘B’	66
‘Z’	90
‘a’	97
‘b’	98
‘z’	122
‘0’	48
‘9’	57

Constant Variables

- A variable that is declared and assigned a value that can never be modified during the execution of a program.
- Reserved work “const” indicates a constant variable.
- THIS IS NOT THE SAME AS A MACRO CONSTANT!

```
const int val = 4;
```

Working with Variables

Printing Variables

- In order to print a variable value, we must instruct the printf function on how to do this:
  - 1. Specify the format of the variable
  - 2. The variable name to print

```
printf("The final values are %d and %lf \n",var,y);
```

Printing Variables

Format Specifier	Data Type	description	Syntax
%d	int	To print the integer value	printf("%d",<int_variable>);
%f	float	To print the floating number	printf("%f",<float_variable>);
%lf	double	To print the double precision floating number or long float	printf("%lf",<double_variable>);
%c	char	To print the character value	printf("%c",<char_variable>);

Static Memory (Stack Space)

- Every component typed in C code must be stored somewhere...
  - It just isn't magically memorized.
- Every C program has static memory.
- Static memory is a storage unit that allows components (such as variables) of a C program to be stored for later use (during execution times).
- The stack space also contains any code statements written from your file.
- This static memory is determined when a C file is compiled.
- Static memory CANNOT change in size.
  - You get what you get and don't get upset...
- Static memory is also known as the ***Stack Space***.

Why is it called Stack Space?

- If we were to draw it out by hand, it will look like a stack of plates.
- The plates are adjacent to each other
- Each plate holds an item.
  - e.g., variables/values

Stack Space Visualization

```
1  // Simple C program to demonstrate variables
2
3  // Header file for input output functions
4  #include <stdio.h>
5
6  // main function -
7  // where the execution of program begins
8  int main()
9  {
10
11 printf("Welcome to the variable demonstration pro
12 // Some variable declarations
13 int var;
14 var = 32;
15 double y = 1.34;
16 var = -5;
17 y = 3.2;
18 printf("The final values are %d and %lf \n",var,y
19
20     return 0;
21 }
```

Print output (drag lower right corner to resize)

Stack Heap

main

var
0xFFF000BCC
int
?
0x????????
0xFFF000BCC: 0x?? ????????
0xFFF000BCD: 0x?? ????????
0xFFF000BCE: 0x?? ????????
0xFFF000BCF: 0x?? ????????

y
0xFFF000BD0
double
?
0x????????????????
0xFFF000BD0: 0x?? ????????
0xFFF000BD1: 0x?? ????????
0xFFF000BD2: 0x?? ????????
0xFFF000BD3: 0x?? ????????
0xFFF000BD4: 0x?? ????????
0xFFF000BD5: 0x?? ????????
0xFFF000BD6: 0x?? ????????
0xFFF000BD7: 0x?? ????????

Acknowledgements

Slides adapted from Dr. Andrew Steinberg's COP 3223H course