CS 240 Programming in C

Introduction

Jan 24, 2023

Course Description

C programming for programmers with prior knowledge of some high-level language (e.g., one semester of programming in Java). The course treats C as a machine-level language and as a general-purpose language; it covers number representation, masking, bitwise operations, and memory allocation, as well as more general topics such as dynamic data structures, file I/O, separate compilation, program development tools, and debugging.

Course Material

- Course content will be posted on the URL:
  `https://cs.umb.edu/~aaditya/`
- Check Blackboard and read your email regularly
- For discussion we will use piazza:
  `https://piazza.com/umb/spring2023/cs2402`
- Course Policies:
  `https://cs.umb.edu/~aaditya/course-policies.html`
- Office Hours: TBA, Email for appointment
  aaditya.tamrakar001@umb.edu
- Homework and Quiz dates will be announced on the course webpage.

About C programming language

- C is a general-purpose, low-level programming language that was developed in the early 1970s (43 years ago) by Dennis Ritchie at Bell Labs.
- It is a procedural language, which means that it follows a structured approach to program design.
- C is widely used for system programming, such as operating systems (including all of Unix) and embedded systems, as well as for application programming. It is also a popular language for learning programming concepts and techniques, as it has a relatively small and simple set of keywords and operators.
- One of its most notable features is its use of pointers, which allow for direct memory manipulation. C is also known for its efficiency, as it allows for low-level access to computer resources. The C standard library provides functions for input/output, string manipulation, and other common tasks.
- C++ and C are also derived from C.

ANSI C and ISO C

- ANSI - American National Standards Institute
- ISO - International Organization for Standardization
- ANSI and ISO are standards organizations that developed a standard specification for C in the late 1970s and 1980s.
- During that time, versions of C were implemented for a wide variety of mainframe computers, minicomputers, and microcomputers, including the IBM PC, as its popularity began to increase significantly.
- The C standardization process aimed to produce a superset of KR C, incorporating many of the subsequently introduced unofficial features. C89 is supported by current C compilers, and most modern C code is based on it. After the ANSI/ISO standardization process, the C language specification remained relatively static for several years. In 1995, Normative Amendment 1 to the 1990 C standard was published, to correct some details and to add more extensive support for international character sets.

C vs modern programming languages

- C is a procedural language, while modern languages often have object-oriented and functional programming features.
- C is considered a "low-level" language, while modern languages are "high-level".
- C has a relatively small and simple set of keywords and operators, while modern languages have more built-in functionality.
- C requires manual memory management, while modern languages often have automatic memory management.
- C is widely used for system programming, while modern languages are often used for web development, data analysis, and other modern applications
- C is considered efficient and fast, while modern languages often have a trade-off between ease of use and performance.
- C requires more lines of code to accomplish a task, while modern languages often provide more abstraction and built-in functionality.
- C is more vulnerable to security issues like buffer overflow and pointer manipulation, while modern languages often offer more security

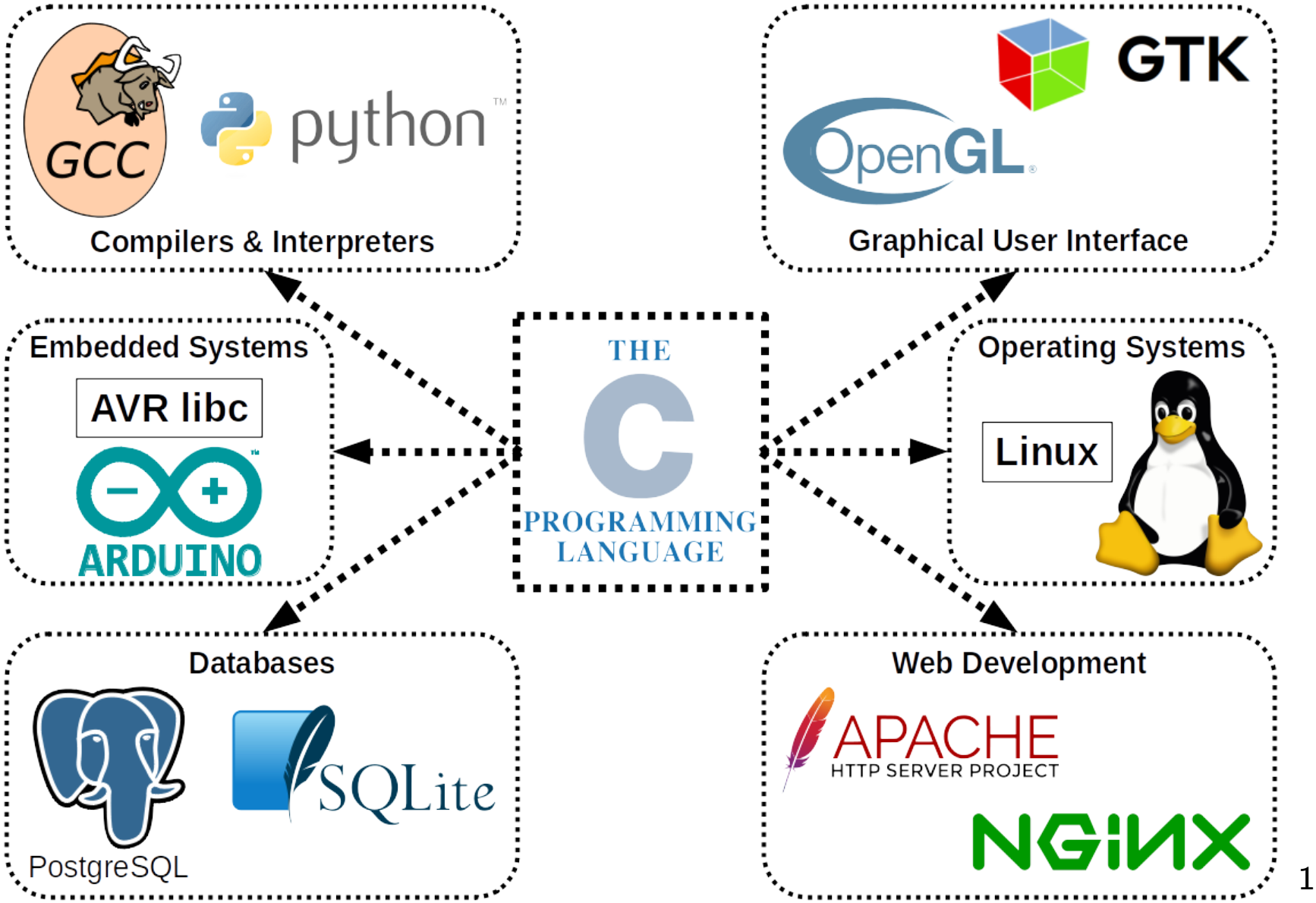
Reasons to learn C

- C is a foundational language: C is considered a foundational language, and learning it can help one understand how computers work at a low level. Understanding the basics of C can help one understand other programming languages and concepts more easily.
- System programming: C is widely used for system programming, such as operating systems, embedded systems, and device drivers. These are areas where Python cannot be used and if one wants to work on these areas, learning C is essential.
- Performance: C is considered to be very efficient and fast, which is why it's still used in performance-critical applications like gaming engines and video editing software.

Reasons to learn C

- C libraries: C has a large number of libraries and frameworks that can be used to extend its functionality and capabilities. These libraries can be used in other languages as well, and learning C can give one access to a wealth of existing code.
- Industry demand: C is still in high demand in industry, especially in the areas of system programming, embedded systems, and IoT. It is still a widely used language in companies and many job opportunities are available for C programmers.
- Low-level understanding: C provides more control over the hardware, and learning C can give a deeper understanding of how computers and other devices work, which can be useful in many fields such as cyber security, firmware development, and more.

Few applications that are written in C



[1]

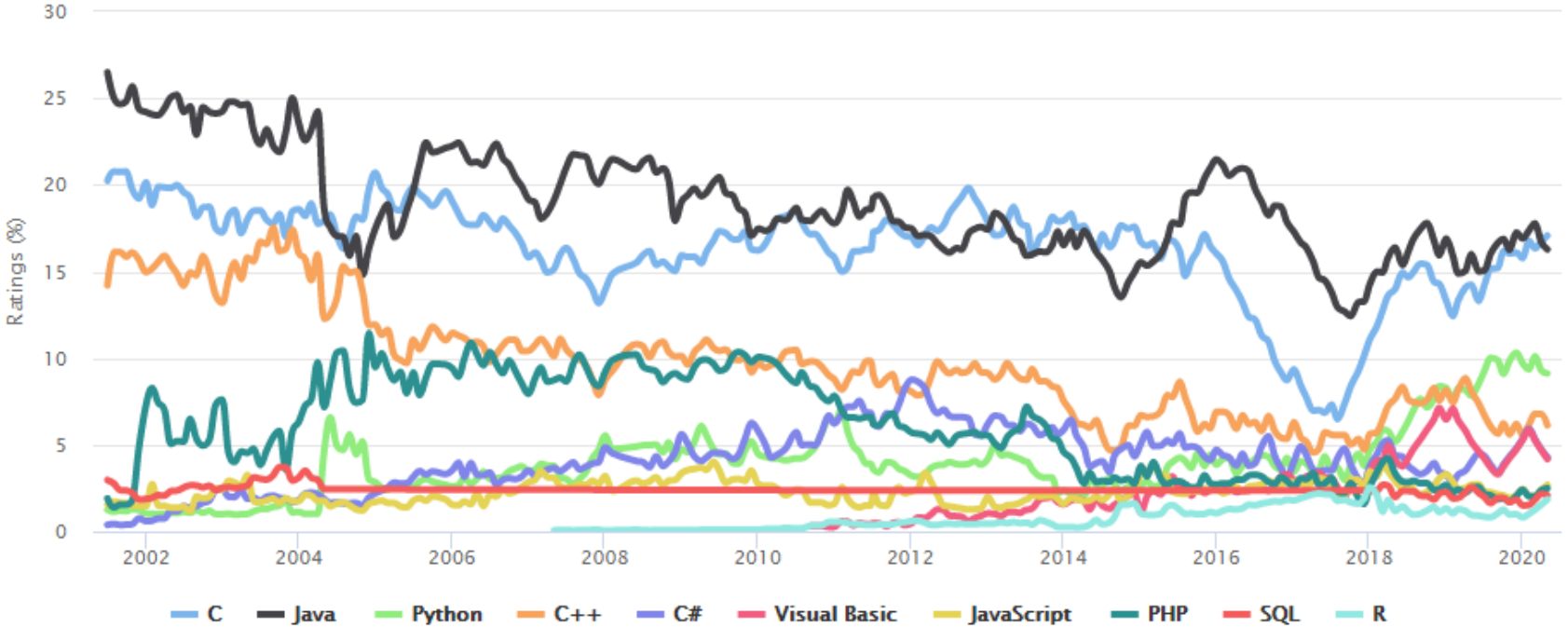
[1] Source: Wikipedia

Few applications that are written in C

- Operating Systems: Many operating systems, including Windows, Linux, and macOS, are written in C or have parts of their kernel written in C.
- Database systems: MySQL, SQLite, and Redis are widely used to store and manage data for websites and other applications.
- Compilers and Interpreters: GCC and Python interpreters are written in C.
- Embedded systems: C is widely used to write firmware and software for embedded systems such as microcontrollers, routers, and IoT devices.
- Graphics and image processing: C is widely used in image and video processing software, libraries, and frameworks such as OpenCV.

Popularity of programming languages

The TIOBE index graph, showing a comparison of the popularity of various programming languages



[2]

[2] Source: Wikipedia

Basic C program structure

Structure of C Program

	1	`#include <stdio.h>`	Header
	2	`int main(void)`	Main
BODY	3	`{`	
	4	`printf("Hello World");`	Statement
	5	`return 0;`	Return
	6	`}`	

3

[3] Source: Geeksforgeeks

Reserved Words

C89 has 32 reserved words, also known as **keywords**, which are the words that cannot be used for any purposes other than those for which they are predefined:
auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

Compiling a C program

Compiling in C refers to the process of converting the source code written in C programming language into machine code that can be executed by a computer. The machine code is a set of instructions that the computer can understand and execute.
The process of compiling C code typically involves the following steps:

- The C compiler reads the source code file(s) and converts it into an intermediate form called object code. The object code contains machine instructions, but it is not yet in a form that can be executed directly by the computer.
- The object code is then passed through a linker, which combines the object code with any necessary libraries and creates an executable file. The executable file contains machine instructions that can be executed directly by the computer.
- The executable file is then ready to be run on the computer.

Compiling a C program

- To compile C code on a command-line interface, one typically uses the GCC compiler. The basic command to compile a C program is "gcc program.c" (without quotes), where "program.c" is the name of the C source code file. This will produce an executable file named "a.out" in the current directory, which can be executed by typing "./a.out" on the command line.
- To specify a different name for the executable file, one can use the "-o" option followed by the desired name, for example, "gcc program.c -o program" will produce an executable named "program".

Development environment setup

- Code Editor: https://code.visualstudio.com/
- GCC:
  - Windows: https://code.visualstudio.com/docs/cpp/config-mingw
  - Mac: https://code.visualstudio.com/docs/cpp/config-clang-mac
  - Linux: https://code.visualstudio.com/docs/cpp/config-linux

HelloWorld.c

Source Code:

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

Commands:
Compile: gcc HelloWorld.c -o hello
Execute: ./hello

Resource Links

- https://en.wikipedia.org/wiki/C_(programming_language)
- https://code.visualstudio.com/docs/cpp/config-mingw
- https://www.geeksforgeeks.org/c-language-introduction/