C++ONLINE

MATHEW BENSON

C++ IN THE DEVELOPING WORLD

WHY IT MATTERS

2024

ABOUT ME

- I like C++!
- I am a Graduate Computer Scientist(since 2007)
- I have been working with and researching on computers and programming and how to practically apply it well for several years, still learning and enjoying the journey.
- I am passionate about energy efficient computing and making the most out of older hardware, researching different platforms, tools and how to bridge different worlds and peoples together.

CONTACTS

- Twitter : @bensonorina
- mastodon : @mathewbenson.hachyderm.io
- email : benson.orina@live.com
- Github : github.com/mathewbensoncode
- Discord : #CppAfrica -> mathewbensonke

AGENDA

- The talk is an exploration of the importance of C++ as a language for writing efficient code for older hardware which is easier to access in the developing world.
- It will also look into the C++ language as a platform for education about software due to the large amount of “real world” software that has already been written in the language.
- It will also touch on tooling and resources for learning & programming C++ in resource constrained situations.

ALTERNATIVE AGENDA

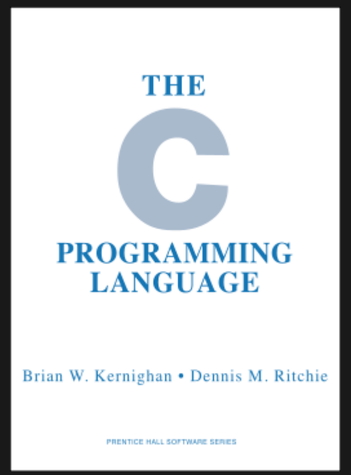
- Respectfully Disagree with The White House
- Hopefully, Prove The White House Wrong 😁

DEFINITIONS

C++

- What is C++?
- It is difficult to answer this without talking about C.

C



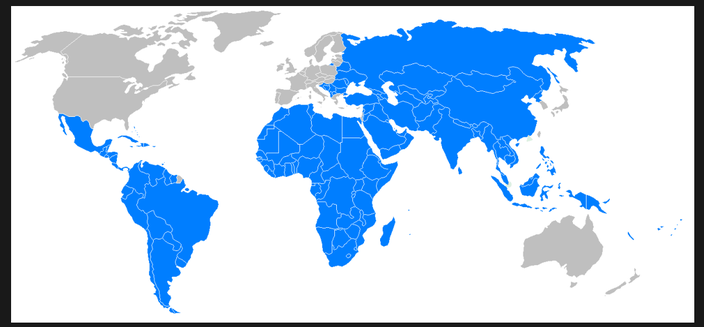
- As you may be aware, C++ is an evolution of the C programming language. This evolution is analogous to a journey.
- C was necessary as a step up from assembly.
- It was born of a need to better express programming intent
- Works with low level constructs like memory and pointers

++

- C++ was a necessary step up from C.
- Originally Called C with Classes
- The Core Concept here is Types of data(Classes)
- I Argue that this refers to the Continuous Development of the Language One Step At A Time.

BUT C++ IS NOT C

- C++ has its roots in C
- It offers Compatibility with C
- There is no language called C/C++
- C++ is an independent language



DEVELOPING WORLD

- The Wikipedia definition of “Developing World” is from the perspective of countries, income levels and access to money.
- I will take a slightly different trajectory in this talk.
- Let’s First Discuss what Developed Means in Terms of Computers

DEVELOPED COMPUTING

- To have enough Computing Power(Cores, Memory, Monitors, etc)
- To have Internet Access That doesn’t *FEEL SLOW*
- To Have uninterrupted Power(Electricity)
- To Have uninterrupted connectivity. (Its available when the User Needs It)
- To Have adequate KNOWLEDGE, to make the most of the computing resources available
- To have All of the Above at a *COMFORTABLE COST*

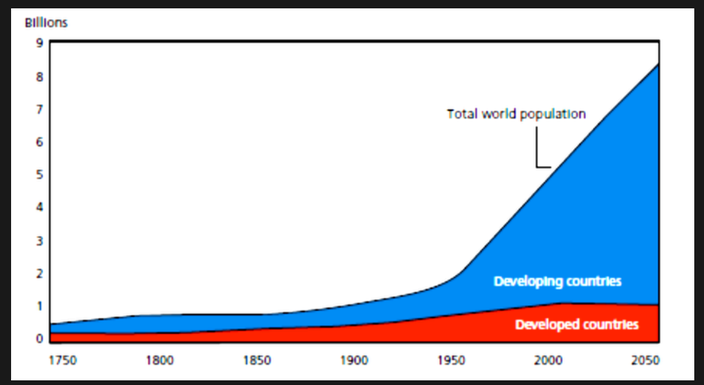
DEVELOPING

- The process of “Growing Closer To” / “Moving Towards” an envisioned, state that is ideal and comfortable -> “DEVELOPED”
- So in Terms of Computers, I mean an experience that is lacking in ANY of the above requirements.
- This Can also Be Looked at as The Development of Our Languages to Better Communicate Our Intents

WORLD

- My emphasis is on a world, not a country.
- This means any person, group, region.
- In reality, there are few places that have that ideal developed experience.
- I Believe that there are not many places with the perfect set of circumstances
- There is also a tendancy for there to be decrease in the above factors the further one goes from the “developed” areas

THERE IS A LARGE POPULATION THAT CAN’T BE IGNORED



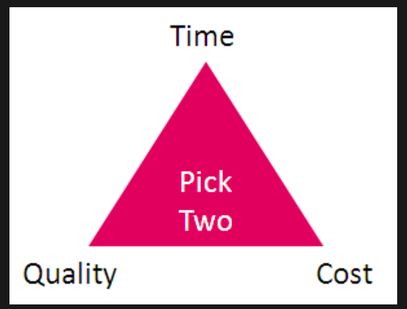
- With a large number of people with less to spend on computer hardware, energy and connectivity, access to computing must come at a lower cost.
- This represents a significant population of the World.

LOWER COST HARDWARE

- As *TIME* has progressed, Technology has improved to make SMALLER electrical components.
- Meaning that it takes LESS POWER To do the SAME tasks as before.
- It may also mean, MORE Tasks with the SAME POWER Levels.
- This Technological Improvement does come at quite a significant cost, Meaning that Newer Computer Hardware Costs More, out of Reach for many people.
- However, This Technological Improvement, does push down the cost for the “Previous Generations” of Hardware,

EFFICIENT CODE FOR BUDGET/OLDER HARDWARE

THE RIDDLE



- Computers are Very Expensive
- A Great Computing Experience is Very Expensive
- A Large Population of the World Struggles to meet this Expense
- Alot of Software is Developed to A Target Market with Lots of Money To Spend(e.g. Corporates, Governments)
- With Better Written Software, Older Hardware Has A Lot of Potential

HOW COMPUTING IS DELIVERED AT A LOWER COST

- One way is that computers come in a different form, namely budget mobile phones.
- The second way is that older, “Obsolete” computers from the “Developed” World find their way to the “Developing” World.
- To hit the lower price points on “Newer Budget” Offerings, manufacturers will likely use older technology rebranded as new.
- However, Alot of Code is written to target Today’s latest and greatest

DIFFERENCE BETWEEN THE CONNECTED AND THE UNCONNECTED

- Chromebooks, have been deployed in some countries as a solution to low spec, mass deployment of computing resources.
- This however comes with the precondition of connectivity.
- This solution cannot work in the “Developing” World.
- Mobile Phones on the other hand, do hold some potential.

SOLVING THE RIDDLE WITH SOFTWARE

- In the “Developing World”, the computing systems available to most users is usually underpowered or very power inefficient.
- Therefore Software Performance is Crucial.
- It is important to always think about the low end, because this is where the people that need it the most will lie.
- It is also where scale matters, for example, the moment you are dealing with large deployments, e.g. to schools, governments,
- Projects to enable access to Computing will usually fail because of the COST of ADEQUATE computing.

SOFTWARE EVOLUTION

- Just as Hardware has gradually evolved, Our software is also improving step by step.
- Optimizations that may not exist when a certain generation of hardware was first used, come up with time.

WHY NOT WEB AND MANAGED LANGUAGES

- The Cost of using the browser or managed languages may be too heavy on older hardware
- On Modern Hardware, the differences may barely be felt giving the illusion that it is fine
- The Safety Argument Doing The Rounds I presume prioritises “Safety” at the expense of “Perfomance”

HOW DOES C++ SOLVE THIS PROBLEM?

BY DESIGN

- It is one of the core fundamentals of the language. The “Zero OverHead Principle”
- There are opportunities to avoid unnecessary repetition

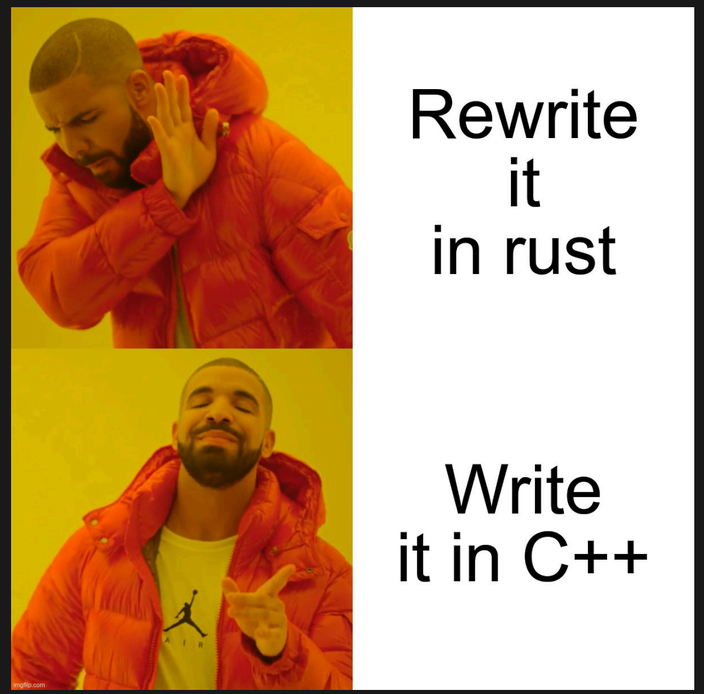
STACK BASED

- We have a lot more control over what we do with our programs.
- It is easier to scale across computing cores when you are in control of the program structure.
- No Third-Party(Garbage Collector/Manager) Between Our Code and Hardware -> Better Perfomance

COMPILE TIME COMPUTING

- The less work that will be done on the resulting client PC, the better. Meaning, the less translation to machine code, the better. (I am aware that there are ways to “precompile” many Interpreted languages, but it is not the natural order of things.)
- Constexpr support. With more done on developer machines, only the necessary run time computations need be done on the “client” machines and this has major potential to unleash power in areas that had not been considered.
- The stack based memory model for computing is better idiom. Being in control of what is happening on the stack is a more natural way to think about computing.
- With C++, “Memory Management” Functionality Offered by Managed Languages, is Perfomed at Compile Time rather than Run-Time

EVOLUTION FROM C/ BACKWARDS COMPATIBILITY



- The backwards compatibility with C is crucial since a lot of critical sofware at the lower level is actually written in C.
- To allow chances for improvement of code by more developers in a safer way; We can use safer tools to experiment, optimize and interact with our systems.
- It may be the safest way to access system level features of our operating systems in a safer way. Alot of the software and libraries commonly used in the tasks you would want to use are written in it.
- For example libraries like opengl, vulkan, direct3d that provide access to hardware graphics capabilities require the use of C level constructs, but when wrapped in C++ containers, types, with the use of RAII

LEARNING/TEACHING C++

- Now that we have talked a little about the “Why” I would like to touch on the “How”
- I am not a teacher of the C++ language, just someone who happens to have learnt through effort.

OH, THE HUMANITY, THE DIVERSITY

- This talk is really not about statistics, facts, figures and measurements. It was born of my reflections after a long hard journey working with software both as a scientist, developer and as a user.
- So I may not have some hard numbers, but I seek to appeal to our reflection on an individual basis on these issues.
- I believe we all people are conversant with the concept of our diversity.
- This diversity is visible in our different personalities, races, colors, genders, religious views, talents, etc.

COMPARISON TO HUMAN LANGUAGE

- C++ is a language, So Let’s Compare how we learn our Human Languages.

ACCORDING TO THE FAQ SECTION ON LANGUAGE ACQUISITION ON THE WEBSITE FOR THE AMERICAN LINGUISTIC ASSOCIATION

FAQ: Language Acquisition

How do children acquire language? Do parents teach their children to talk?

No. Children acquire language quickly, easily, and without effort or formal teaching. It happens automatically, whether their parents try to teach them or not.

Although parents or other caretakers don't teach their children to speak, they do perform an important role by talking to their children. Children who are never spoken to will not acquire language. And the language must be used for interaction with the child; for example, a child who regularly hears language on the TV or radio but nowhere else will not learn to talk.

Children acquire language through interaction - not only with their parents and other adults, but also with other children. All normal children who grow up in normal households, surrounded by conversation, will acquire the language that is being used around them. And it is just as easy for a child to acquire two or more languages at the same time, as long as they are regularly interacting with speakers of those languages.

HOW ARE WE DESIGNED TO LEARN LANGUAGE

- When Children are born they have a natural ability to pick up language, usually the language they interact with first.
- Children Acquire Language by interaction with the language through talking with adults and other children.
- There are certain stages that are similar in acquiring language.
- This learning is gradual and happens step by step.
- Adults Help to Improve the Learning of Children by Stepping Down A level Closer to Children Through “Baby Talk”

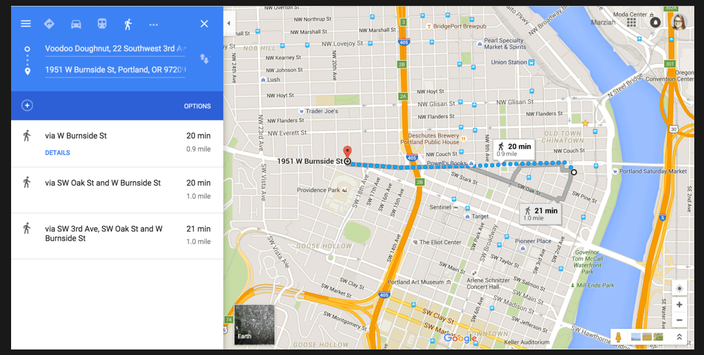
HOW DOES THIS MAP TO LEARNING C++

- We need to bring the human interaction into it.
- Through our Communities, Our Work Environments
- I believe this to be one of the benefits of our C++ Conferences, User Groups, Discord Servers, etc.
- The more we speak about code, interract with our code, read our code the better we get at using our computers

RE-EVALUATING “WHY” WE USE COMPUTERS AND PROGRAMMING

- As with many things in life, many people use(learn) computers and programs due to reasons that often conflict with the intended design of said computers and programs.
- This is different for each individual
- So we all have our different starting points

ROADMAP FOR LEARNING C++

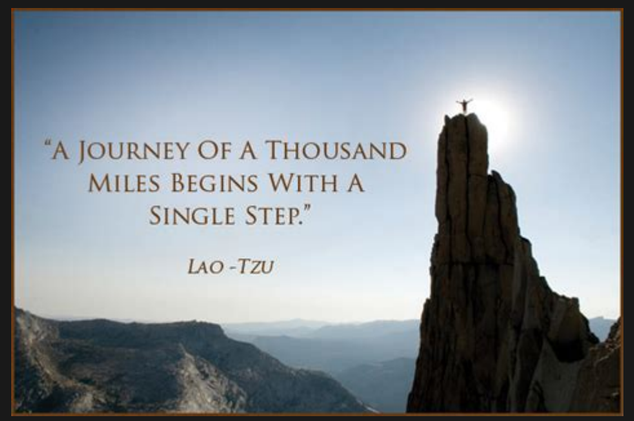


- There may not be a one-size-fits-all answer to this for everyone as we are all diverse
- You may need to try different teachers, books, resources & techniques before you find the right mental model for your unique mind, talent, etc.
- We are all learning, just that some of us have been learning for longer.

“C” THE STORY

- One of the greatest, in my opinion, reasons behind my preference for C++ in particular is that it is part of a rich history.
- Like a movie that is part of a series of movies where the story just couldn’t fit in one movie length e.g Lord Of the Rings, The Star Wars Story, etc.
- Explore why things are the way they are, many features come about as a result of rich discussions, arguments, testing, trial and error

ONE STEP AT A TIME



- "The Journey of a thousand miles begins with a single step"
- Keep a Reference Close At Hand, as you would use a map. (http://cppreference.com)
- The Heart of the Plus-Plus part of C++

TOOLING & RESOURCE TIPS

FREE LANGUAGE LEARNING RESOURCE

HTTPS://CPPREFERENCE.COM

C++ reference

C++11, C++14, C++17, C++20, C++23, C++26 | Compiler support C++11, C++14, C++17, C++20, C++23, C++26

Language
- Keywords – Preprocessor
- ASCII chart
- Basic concepts
  - Comments
  - Names (lookup)
  - Types (fundamental types)
  - The main function
- Expressions
  - Value categories
  - Evaluation order
  - Operators (precedence)
  - Conversions – Literals
- Statements
  - if – switch
  - for – range-for (C++11)
  - while – do-while
- Declarations – Initialization
- Functions – Overloading
- Classes (unions)
- Templates – Exceptions
- Freestanding implementations

Standard library (headers)

Named requirements

Feature test macros (C++20)

Language support library
- Program utilities
- source_location (C++20)
- Coroutine support (C++20)
- Three-way comparison (C++20)
- Type support
- numeric_limits – type_info
- initializer_list (C++11)

Concepts library (C++20)

Diagnostics library
- exception – System error
- basic_stacktrace (C++23)

Memory management library
- unique_ptr (C++11)
- shared_ptr (C++11)
- weak_ptr (C++11)
- Memory resources (C++17)
- Allocators – Low level management

Metaprogramming library (C++11)
- Type traits – ratio
- integer_sequence (C++14)

General utilities library
- Function objects – hash (C++11)
- Swap – Type operations (C++11)
- Integer comparison (C++20)
- pair – tuple (C++11)
- optional (C++17)
- expected (C++23)
- variant (C++17) – any (C++17)
- String conversions (C++17)
- Formatting (C++20)
- bitset – Bit manipulation (C++20)
- Debugging support (C++26)

Strings library
- basic_string – char_traits
- basic_string_view (C++17)
- Null-terminated strings:
  byte – multibyte – wide

Containers library
- vector – deque – array (C++11)
- list – forward_list (C++11)
- map – multimap – set – multiset
- unordered_map (C++11)
- unordered_multimap (C++11)
- unordered_set (C++11)
- unordered_multiset (C++11)
- Container adaptors
- span (C++20) – mdspan (C++23)

Iterators library

Ranges library (C++20)

Algorithms library
- Execution policies (C++17)
- Constrained algorithms (C++20)

Numerics library
- Common math functions
- Mathematical special functions (C++17)
- Mathematical constants (C++20)
- Basic linear algebra algorithms (C++26)
- Numeric algorithms
- Pseudo-random number generation
- Floating-point environment (C++11)
- complex – valarray

Date and time library
- Calendar (C++20) – Time zone (C++20)

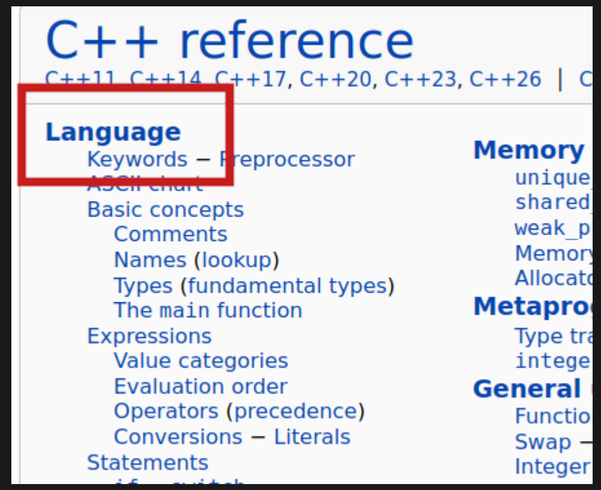
Localization library
- locale – Character classification
- text_encoding (C++26)

Input/output library
- Print functions (C++23)
- Stream-based I/O – I/O manipulators
- basic_istream – basic_ostream
- Synchronized output (C++20)
- File systems (C++17)

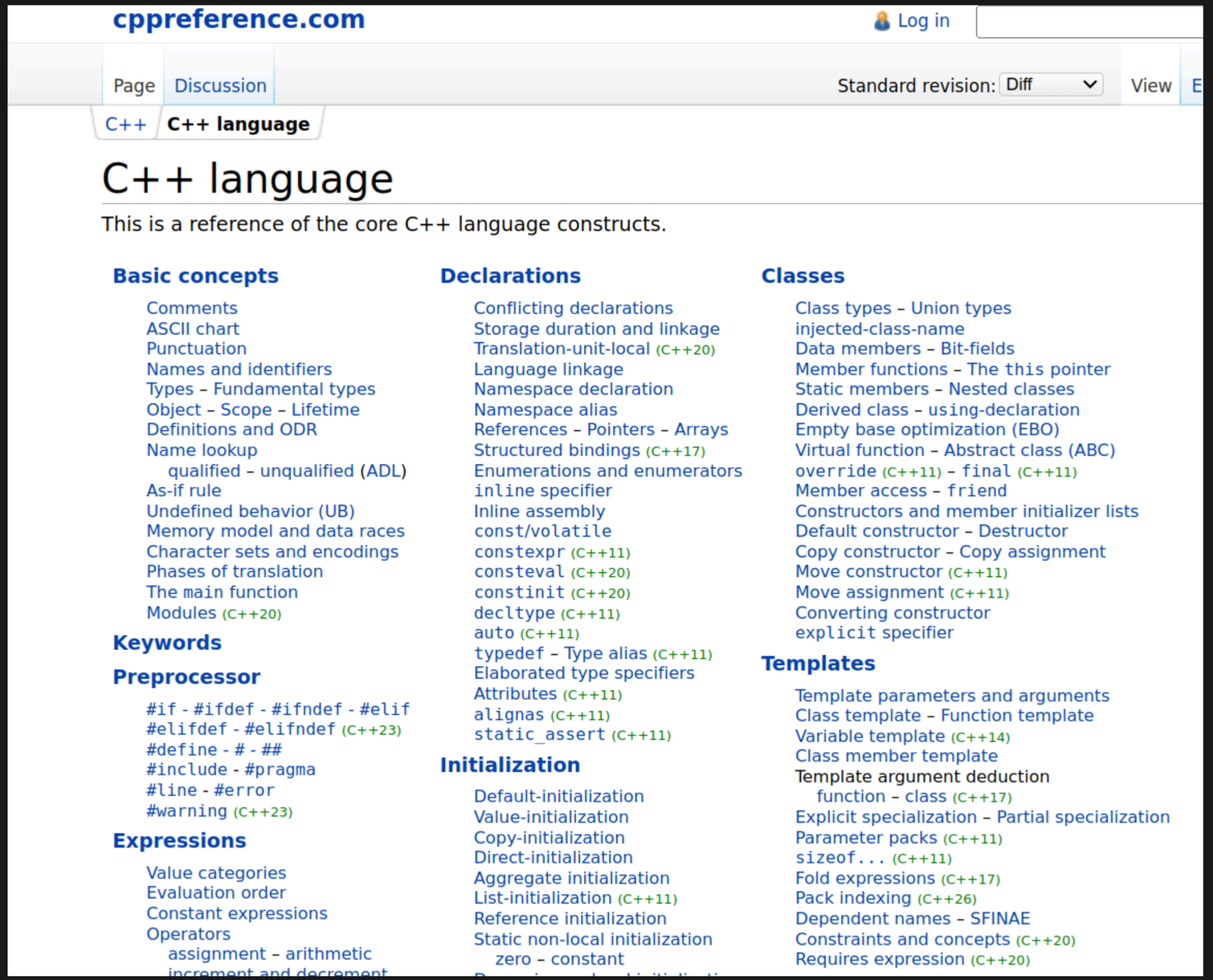
Regular expressions library (C++11)
- basic_regex – Algorithms
- Default regular expression grammar

Concurrency support library (C++11)
- thread – jthread (C++20)
- atomic – atomic_flag
- atomic_ref (C++20) – memory_order
- Mutual exclusion – Semaphores (C++20)
- Condition variables – Futures
- latch (C++20) – barrier (C++20)
- Safe Reclamation (C++26)

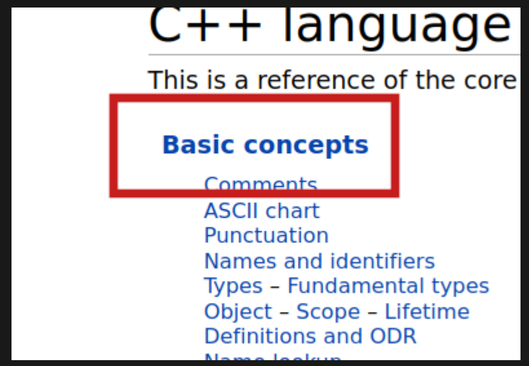
START AT THE TOP LEFT CORNER - LANGUAGE



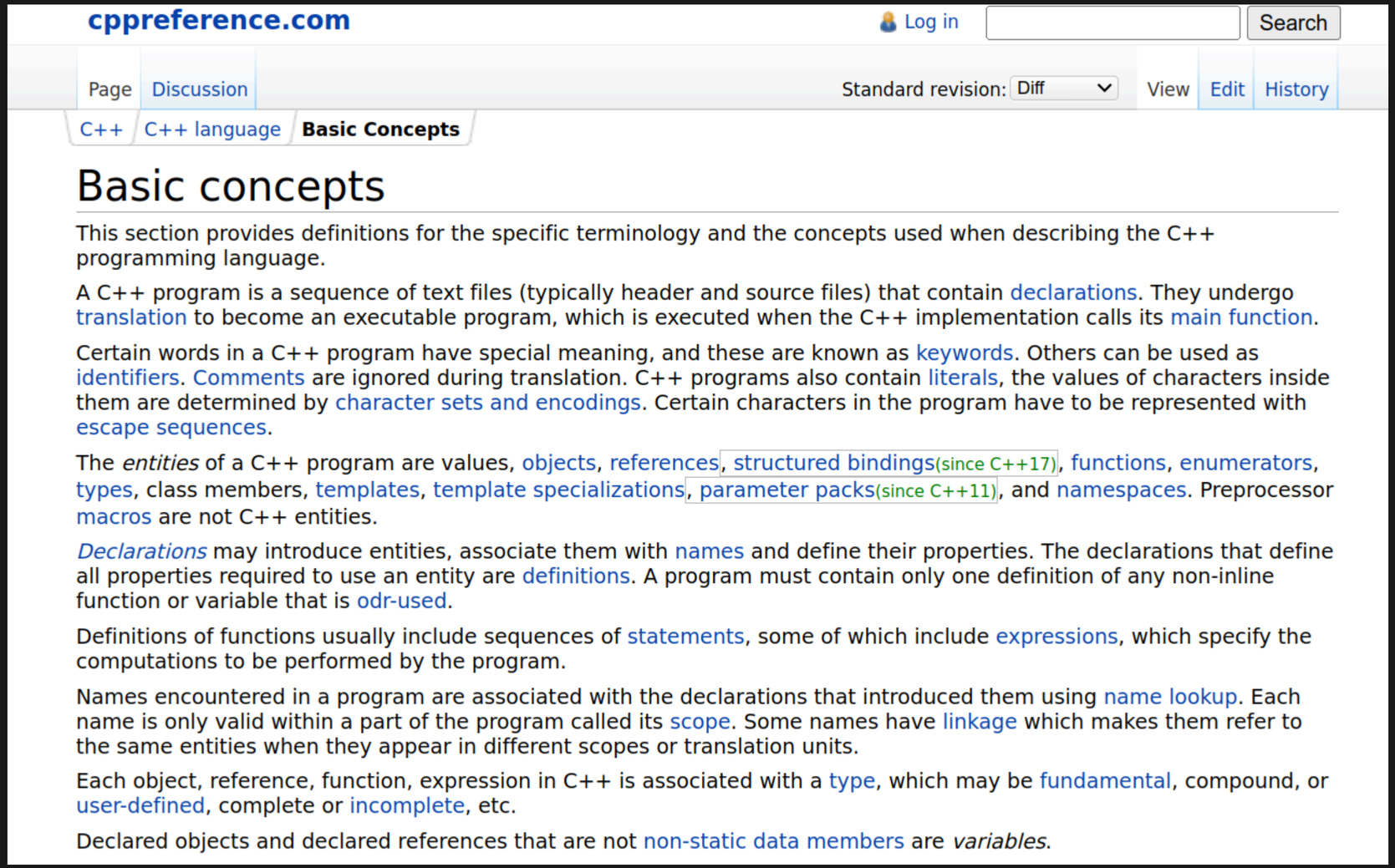
CPPREFENCE - C++ LANGUAGE SECTION



START AS AT THE TOP LEFT CORNER AGAIN

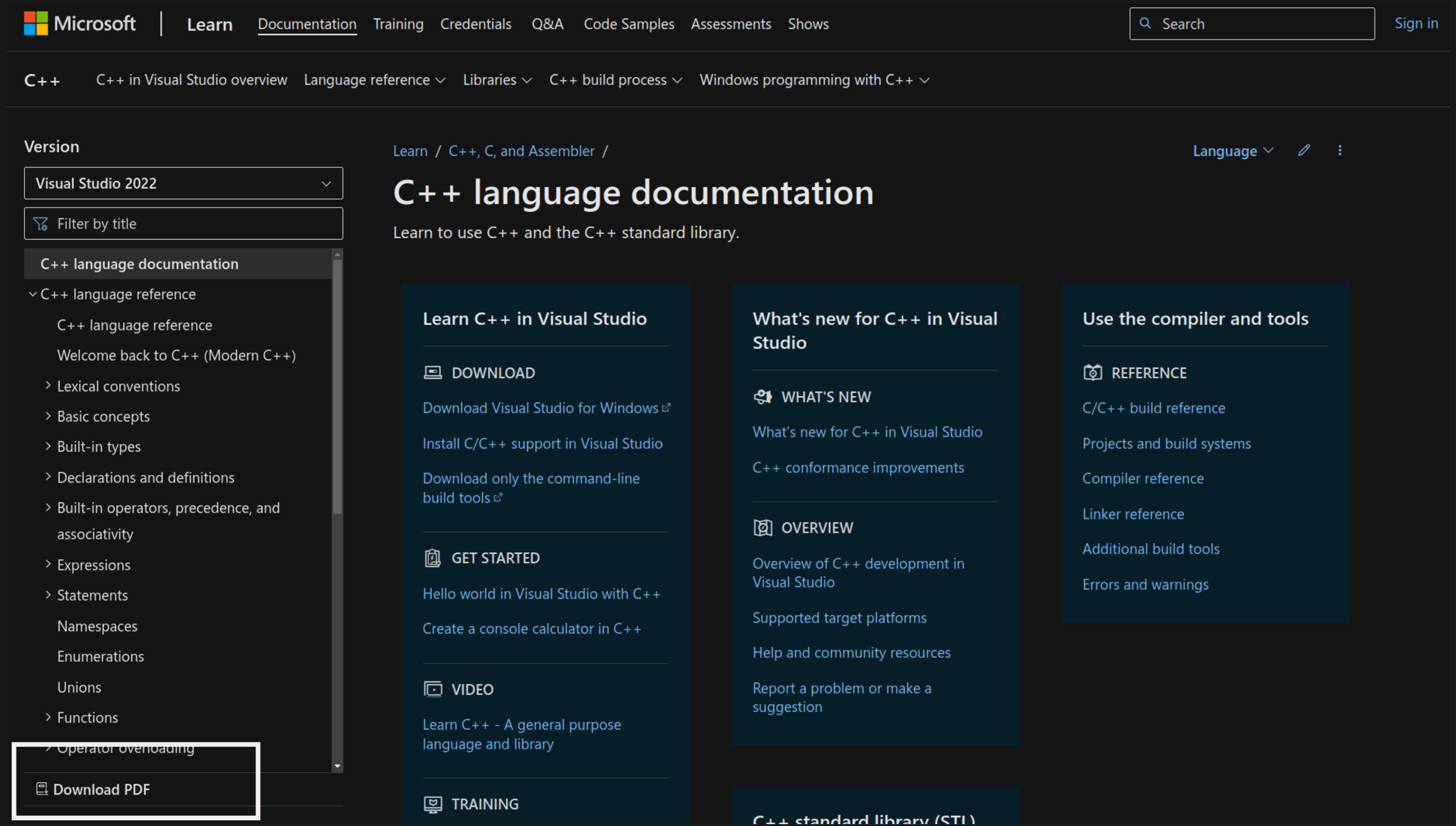


CPPREFERENCE - LANGUAGE BASICS



MSDN FREE C++ DOCUMENTATION

- Downloadable PDF.



IMPORTANT TERMS TO START WITH

- expression
- statement
- variable
- object
- Declaration
- Definition
- Function

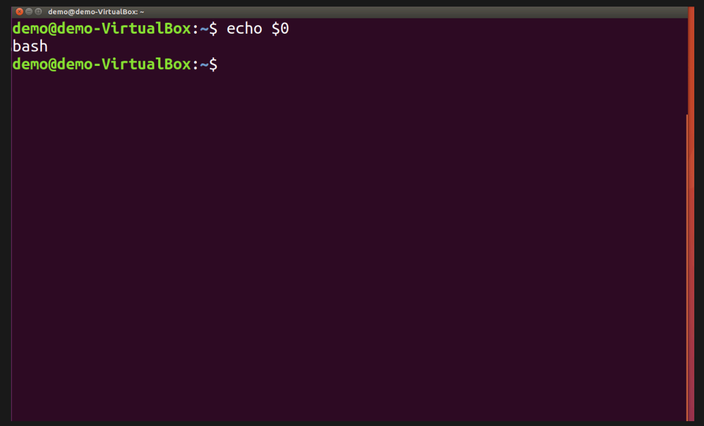
FOR OLDER OR CONSTRAINED COMPUTING SCENARIOS

- To be able to boost interactivity with the Language, there must be access to computers, power, books, connectivity, etc.
- It is important to crawl before you walk and walk before you run.
- In this regard, I think the answer may be to look into the past a little and Look into how computers worked in times past.

DON’T START WITH INTEGRATED DEVELOPMENT ENVIRONMENTS

- When Starting out it may be cheaper, and more effective to avoid relying on IDEs when starting to learn.
- In my opinon the IDEs like visual studio, clion and the rest are great when you are already in professional environments and with more powerful hardware, and when time is of the essence in professional environments.
- The value proposition of IDEs really shines once you are aware of what it is they really do for you.

THE TERMINAL



- I believe that it is important to first understand where computing is coming from.
- The Terminal is essential to this.
- Knowing about Plain Text Files, compiler commands, linker commands is essential.
- It is also the lowest cost model for teaching programming to more people, in more situations.
- Learn Bash(Posix Shell) or Powershell. These are invaluable

KEEP AS MUCH AS YOU CAN LOCALLY

- Having all you need on your developer machine is better than always having to rely on resources on the internet. Especially in places with expensive/limited connectivity.
- It is possible to download the entire cppreference.com website locally. It is available as a package on arch-linux.
- Get Reference material like the Pdf of the microsoft developer website C++ documentation on PDF.
- Buy Good Books On C++ When you Can Afford To.

SHARE COMPUTE RESOURCES ON LOCAL NETWORKS WHERE POSSIBLE

- You can have a local network version of compiler explorer.
- This is To Reduce Connectivity Costs and Use the Limited Connectivity for the most Crucial Things.

SSH

- This is one of the benefits of Terminal Based Computing.
- Try To Learn About Vim/Neovim or Emacs
- Or share a single computer with multiple users who may use basic android phones with terminals via ssh-clients. This works with windows PCs.
- I use this personally as my personal machine is Linux based, but always ssh into my wife’s laptop to try things out on windows using neovim on both.

CONCLUSION

- For our Language C++ to thrive and be of greater benefit, Every One of us Must be as welcoming as possible to new users.
- We need the performance benefits of C++ applied to where it matters most i.e. on Older, Cheaper Hardware to make computing more accessible to more people in more places.
- Older Tooling and Software such as the Terminal that would be considered by some to be obsolete can, when applied appropriately be the key to creating opportunities for people who may not have had a chance otherwise.
- The Personal Computer Can Do so much more With Better Code.