accu 2021

VIRTUAL EVENT

Spring ACCU 2021.3.10-13

Pre-Conference tutorials 2021.3.9

2021 CONFERENCE PROGRAMME

Follow @ACCUConf

Tweet #ACCUConf

HEADLINE SPONSOR

Bloomberg

Engineering

Bloomberg is the leading provider of financial news and information. Our R&Ddepartment consists of 3000 engineers working across teams in both London and New York.
We are technology people – when we write code, we like to understand exactly what the hardware is doing. We live and breathe C++, and are concerned about cpu performance, optimizations, memory allocation behaviour, as well as higher-level software engineering principles such as building modular, scalable, reliable and debuggable code. We write a lot of systems ourselves, but have embraced the open source community. The Bloomberg Terminal is the primary product we are known for – it used to run on custom hardware, but is now a piece of software that runs on a PC – typically connected to 2 or more monitors. Over 315,000 customers pay a monthly subscription to access our real-time data and news, deep analytic functions and trading functionality through our custom browser. You will see many Bloomberg Terminals on most trading floors around the world, and in the offices of influential decision-makers within financial markets.

OTHER SPONSORS/PARTNERS

mosaic

Mosaic Financial Markets is a consulting firm that offers a compelling and disruptive alternative to the traditional consulting model.

We deliver in close partnership with our financial markets clients: changes to their strategy, business processes and technology by providing content-rich expertise in a cost effective manner. Our global network of independent expert consultants enables our clients to access a deep pool of professional knowledge ranging from individual technical specialists to project teams and full consulting solutions.

Mosaic is delighted to support the 2021 ACCU conference. The largest (and rapidly growing) part of our business is helping clients solve hard technical problems, so we welcome ACCU's focus on developing professionalism in programming and improving programming skills. A number of ACCU people are key members of our teams delivering complex projects for our clients. We are always growing our expert consultant network and want to hear from exceptional technical professionals whose outstanding track records attest to their proven expertise and problem solving skills.

To learn more about what we do please get in touch: **www.mosaic.fm** or email **info@mosaic.fm**

undo

Undo is the leading Software Failure Replay platform provider for engineering teams building complex systems. Its products – UDB and LiveRecorder – help reduce the Mean Time-To-Resolution (MTTR) of software defects by eliminating the guesswork in bug diagnosis.

Built for mission-critical software, Undo's products are trusted by the world's largest technology firms to quickly resolve defects in complex applications across all phases of the software development life cycle, allowing them to accelerate software delivery, and more quickly resolve customer issues.

With offices in Cambridge, UK and San Francisco, CA, Undo's solutions are used by thousands of software engineers across leading technology companies including SAP, Juniper, Cadence Design Systems, Actian and Mentor (a Siemens business).

C++ on Sea

#include <C++>
www.includecpp.org

2021 CONFERENCE WELCOME

Welcome to ACCU 2021!

ACCU Conference is the annual conference of the ACCU membership club, but open to any and all who wish to attend. The ACCU Conference has a history founded in studying and evolving C and C++ – many of its members continue to have active roles in the C and C++ standards bodies. The ACCU Conference has always had C++ at it's core, but has, from the outset, always been about programming languages, programming tools and techniques, and programming processes. ACCU Conference is a conference by programmers for programmers about programming. Or you could substitute software development for programming and it would mean the same thing. Many ACCU attenders, and ACCU members are software developers/programmers working as employees in organisations, many are consultants, many are contractors, all are in the vanguard of thinking about programming, software development, and how to do it better.
The ACCU Conference is rightly proud of the ACCU (the organisation) banner "professionalism in programming".
Along with its C++ core content, ACCU Conference always has sessions relating to languages such as C#, D, F#, Go, Haskell, Java, Kotlin, Lisp, Python, Ruby, Rust, and Swift. Also there are always sessions on TDD, BDD, and how to do programming right. It isn't just about the programming languages, it is also about the tools and techniques of programming.

The core of the ACCU Conference is about developers who want to share their experience in the form of presentation and interactive workshops with other developers. Even though the conference in 2020 had to be cancelled because of the COVID-19 pandemic, many speakers are happy to give their presentations in 2021.

So we have our usual mix of keynotes, sessions, quickies and lightning talks.

Many thanks to all the session presenters for submitting, making preparations, and running the session. Many thanks as well to all attenders for being online and making the conference under these difficult conditions work.

A special thanks to the sponsors and exhibitors: they are as much attenders as anyone. Hopefully everyone will get a chance to talk to everyone.

The programme presented in this document is the way we planned the programme, however there may have to be last minute changes. If there are, they will be tweeted via hashtag #ACCUConf, and the webpage **https://flame.firebird.systems/archer-yates/ACCU2021/MyProgrammes#Programme.ItemPage.104.0** will be updated.

Our Code of Conduct - **https://accu.org/conf-menu-overviews/coc_code_of_conduct/** - is of course valid for this online conference. In case of any concerns or problems, please do not hesitate to get in touch directly with the organizers, via the Discord channel.

This is ACCU 2021, enjoy the keynotes, sessions, quickies and lightning talks. Use Twitter with the hashtag #ACCUConf. Chat with all the exhibitors, and with each other. ACCU is a conference for conferring!

Felix Petriconi
Conference Chair

Ready to build the platform of tomorrow right now?

We're building the world's most trusted information network for financial professionals. Our 6,000+ engineers, developers, and data scientists are dedicated to advancing and building new solutions and systems for the Bloomberg Terminal and other products in order to solve complex, real-world problems.

bloomberg.com/careers

Architect **on purpose.**

Bloomberg Engineering

2021 CONFERENCE SESSIONS INFO

Tue 09 Mar 10:00-18:00

Title	Oral Session
Presenter	Gail Ollis
Presenter	Kevlin Henney
Presenter	Giovanni Asproni
Presenter	Chris Oldwood
Presenter	Roger Orr
Chair	Gail Ollis
Guest Presenter	Kevlin Henney

0043

ACCU 101: Early Career Day

GAIL OLLIS[1], KEVLIN HENNEY[2], GIOVANNI ASPRONI[3], CHRIS OLDWOOD[4], ROGER M. ORR[5], JON SKEET[6], JEZ HIGGINS[7], ARJAN VAN LEEUWEN[8]

[1] Several universities, Bournemouth, Portsmouth, Open University, United Kingdom [2] Curbralan, Bristol, United Kingdom [3] Unknown, London, United Kingdom [4] Freelance, Cambridgeshire, United Kingdom [5] Freelance, London, United Kingdom [6] Google, London, United Kingdom [7] Freelance software grandad, Birmingham, United Kingdom [8] Unknown at time of submission, Unknown at time of submission, The Netherlands

This is an all-day pre-conference tutorial. But unlike other tutorials, the Early Career Day is exclusively for software developers in their first few years of work, whether as a placement student or a graduate level employee. In the company of others with a similar level of experience, this supportive tutorial will offer you clear and practical guidance in key aspects of your work. Your tutors are wellknown speakers selected for their excellent content and clear delivery. If you are wondering whether the ACCU conference is for you and how it can help you, this is a great way to try out a 'miniconference' designed especially for you by experts. The day is chaired by Dr Gail Ollis, an ACCU conference regular who remembers what it was like to attend her first ACCU conference. After a long career in commercial software development, Gail is now a lecturer and researcher in software development. Think of the Early Career Day as a short course in honing your craft as a software developer, with an experienced lecturer as your course leader and expert presenters as your tutors. The content covers a mix of personal and technical skills, including: * Presentation skills (in person and online) * Software processes and architecture * Coding practices * Code review * Getting help online * Debugging * Deployment The day includes your very own lightning talk session so you can, if you choose, practice sharing your thoughts at a conference. You're warmly invited to take this opportunity to practice with a small, friendly, supportive audience, but participation is absolutely voluntary.

Tue 09 Mar 10:00-18:00

Title	Oral Session
Presenter	Mateusz Pusz

2021

Modern C++ Idioms

MATEUSZ PUSZ[1]

[1] Epam Systems | Train IT, Gdansk, Poland

C++ is no longer C with classes and it never was only an Object Oriented language. C++ is a general purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. If used correctly, it provides hard to beat performance. Such usage requires a good knowledge of C++ templates and Modern C++ Idioms which are much different from commonly known design patterns popularized by GoF book and invented to handle common use cases in pure OO languages like Java or C#. What you will learn: During the workshop, we will refresh and broaden our knowledge about C++ templates and will learn Modern C++ Idioms like Niebloid, EBO, CRTP, Type Erasure, and many more. Crafting those skills will allow us to build powerful tools that are useful in the everyday work of every C++ developer. Experience required: In order to be able to follow the workshop, you should be current with C++ and have some recent experience with writing simple C++ templates. C++11/14 knowledge is suggested but not mandatory. Environment: A laptop with a relatively new C++ compiler. It is recommended to have the latest version of one of the compilers (Visual Studio, gcc or clang).

Tue 09 Mar 10:00-18:00

Title	Oral Session
Presenter	Peter Sommerlad

2021

Good Modern C++ Design and Practices

PETER SOMMERLAD[1]

[1] Better Software: Consulting, Training, Reviews, Wollerau, Switzerland

This workshop is trying to simplify your use of C++. We have many great rule sets to chose from, some partially outdated, like Scott Meyers 3rd edition, some futuristic, like the C++ core guidelines.

While working on the AUTOSAR C++ and new MISRA C++ guidelines I found that many of the guidelines forbid things without giving actual guideline on how to do things and when to deviate.

Also many talks on C++ explain the modern features and show how they work, but only few put things into context and show what to give up and how things combine sanely. I am guilty of that in the past as well, e.g., with my constexpr compile time computation talks at ACCU. This full day workshop is the result of thinking about that. It won't show C++20 feature by feature, but gives a coherent set of practices to improve your design and code using existing standard C++ features where they give you benefits. We will cover the following topics: * designing function interfaces in a way that they are easy to call correctly and hard to call incorrectly * how to report function contract violations (at least 5 different ones) and their individual benefits and liabilities, so you can make a conscious choice. * what parameter passing style and return value style works best under what conditions * how to create (parameter) type wrappers to avoid passing wrong arguments * class design for simple value wrappers to improve function interfaces * mix-in strategies for functionality and operators, so that creating value wrappers is simpler * provide an overview of class styles, e.g., value, manager, oo-bases and show how to select from the rules for special member functions * take a look at the lesser known C++11 feature of ref-qualified member functions and show why and when to use them for your member functions If you are brave enough, bring your own examples that we can look at and discuss where they are perfect and where they could be improved. Otherwise, we will take a look at potential bugs in the C++ standard library design.

Tue 09 Mar 10:00-18:00

Title	Oral Session
Presenter	Adrian Ostrowski
Presenter	Piotr Gaczkowski

2021

Building and Packaging Modern C++

PIOTR GACZKOWSKI[1], ADRIAN OSTROWSKI[2]

[1] Meraki Acoustic, Gdansk, Poland [2] Intel, Gdansk, Poland

There are many ways to build and package your C++ code, each with a different approach to supporting multiple operating systems and toolchains. Integrating third-party components into your software often means having to deal with even more build systems. All of that without even mentioning topics like cross-compilation. How to grasp and handle all of this complexity? In this workshop, we would like to show you the modern tooling that takes a lot of burden out of build management. We'll cover the state of the art utilities such as CMake, Conan, clang-format, Nix, precommit, and Docker. Because cross-platform development should be fun!

Tue 09 Mar 12:00-20:00

Title	Oral Session
Presenter	Sean Parent
Guest Presenter	Sean Parent

2021

Better Code

SEAN PARENT[1]

[1] Adobe, San Jose, The United States of America

A workshop based on the Better Code series of lectures with opportunities to experiment and discuss the ideas presented. Writing code is challenging, especially writing efficient code on large projects. We'll cover types, algorithms, data-structures, runtime polymorphism concurrency, and relationships. Each section provides insight into how to reason about your code and specific techniques to build better code. The prerequisite for this course is a basic understanding of C++. Developers at all levels will learn some new ideas and techniques to improve their code quality, efficiency, and readability. Requirements: Please come prepared to run a C++17 command-line application. [Links to a github repository with starter code will be provided in advance.]

Wed 10 Mar 09:00-10:30

Title	Invited Slot
Presenter	Emily Bache
Guest Presenter	Emily Bache

2021

Keynote: Technical Agile Coaching with the Samman method

EMILY BACHE[1]

[1] Emily Bache, Göteborg, Sweden

Becoming agile as an organization is a journey. It takes time to change attitudes and behaviours. Especially for a larger organization, you need a multitude of coaches and change agents to succeed. My focus is specifically on coaching technical practices and how people write code. Over the years I've tried several approaches and this talk is about the best way I have found so far to promote agility amongst developers. I work daily with teams, ensemble programming in their production code. I also lead a short 'learning hour' practice session. The combination of these two kinds of activity tends to have a powerful effect on a team and the way they behave in their production code afterwards. In this talk I will explain more about the method, which I have named 'Samman Technical Coaching'.

Wed 10 Mar 11:00-12:30

Title	Oral Session
Presenter	Mathieu Ropert

2021

This Videogame Developer Used the STL and You'll Never Guess What Happened Next

MATHIEU ROPERT[1]

[1] *Paradox Development Studio, Stockholm, Sweden*

The STL is sometimes seen as a strange and dangerous beast, especially in the game development industry. There is talk about performance concerns, strange behaviours, interminable compilations and weird decisions by a mysterious "committee". Is there any truth to it? Is it all a misconception? I have been using the STL in a production videogame that is mostly CPU bound and in this talk we will unveil the truth behind the rumours. We will start by a discussion about the most common criticism against the STL and its idioms made by the gamedev community. Then we will see a few practical examples through STL containers, explaining where they can do the job, where they might be lacking and what alternatives can be used. Finally we will conclude with some ideas on how we can improve both the STL for game developers and also how to foster better discussion on the topic in the future. At the end of this talk, attendees should have a solid understanding of why the STL is sometimes frowned upon, when it makes sense to look for alternatives to the standard and most importantly when it does not.

Wed 10 Mar 11:00-12:30

Title	Oral Session
Presenter	CB Bailey
Presenter	Andy Balaam

2021

What does the linker actually do for us?

CB BAILEY[1], ANDY BALAAM[2]

[1] *Bloomberg LP, London, United Kingdom* [2] *OpenMarket, London, United Kingdom*

In this session Andy and CB explore what linkers actually get up to when they pull together your c++ code and libraries to produce an executable. This session is aimed at developers who have some experience with working with a compiled language such as C or C++ and want to know more about the how the last tool in their toolchain works. The session will examine what information object files typically contain and what is required to make a complete executable program out of one or more object files. All of the examples and demonstrations will be using Linux and ELF object files, but the concepts are applicable across most modern environments. We'll define concepts such as _sections_, _symbols_, _relocations_ and look at how code and data are managed in a program. We'll explore how the operating system runs a program and how this shapes what the linker actually does. We'll also explore aspects particular to C++ such as how template instantiations and inline functions are managed. On the way we'll look at tools that can be used for examining object files and executables that let us dispel the mysteries of the linker.

Wed 10 Mar 11:00-12:30

Title	Oral Session
Presenter	Timur Doumler

2021

How C++20 changes the way we write code

TIMUR DOUMLER[1]

[1] *Timur Doumler, London, United Kingdom*

The upcoming C++20 standard is the biggest update in a decade. Its feature set and their impact on how we write C++ will be as large, and possibly larger than that of C++11. In this talk we will look at how new features like concepts, coroutines, and modules will fundamentally change the way we design libraries, the way we think about functions, and even the way we organise and compile our code. We will also mention some long-standing warts in C++ which are finally cured.

Wed 10 Mar 11:00-12:30

Title	Oral Session
Presenter	Sy Brand

Reconfirmed

Dynamic Polymorphism with Code Injection and Metaclasses

SY BRAND[1]

[1] *Microsoft, Edinburgh, United Kingdom*

Dynamic polymorphism in C++ has historically meant virtual functions and inheritance. However, these form only one possible design for solving this problem, and they bring several implications on performance, ergonomics and flexibility. Type erasure is another way to implement dynamic polymorphism, as demonstrated in several talks by Sean Parent and adopted in other languages, such as Rust's trait objects. But implementing type erasing objects which provide ergonomic interfaces in C++ is cumbersome and error-prone, leading to a large family of types and libraries with subtly different semantics and lower adoption rates compared to inheritance. This talk will present a possible future design for interface-based type erasure in C++ that marries the convenience of inheritance to the benefits which it otherwise lacks. It will introduce the code injection and metaclasses facilities which are proposed for inclusion in C++ along with a prototype implementation of the design based on the experimental metaclasses Clang fork.

Wed 10 Mar 11:00-12:30

Title	Oral Session
Presenter	Filip Van Laenen

2021

Drawing for IT Architects

FILIP VAN LAENEN[1]

[1] *Computas, Oslo, Norway*

Ever seen a drawing trying to explain the architecture or the functionality of an IT system, and you couldn't make any sense of it because it was drawn so badly? Do you feel that your drawings are OK, but could probably be improved substantially with a few tricks, but you don't really know what they would be? I've made a lot of drawings over the years. I'm pretty sure not all of them were a success in terms of understandability. But at some point in time, I started reflecting on how my drawings look, and since then, I'm getting comments that apparently, my drawings look good. In this session, I would like to share some of the reflections I make when I see somebody else's drawing, show you some obviously bad drawings and discuss what's wrong with them, and some of the tricks I use to create better drawings. The goal should be that when you leave this session, you've practiced a bit on how you can make your drawing look better, such that the reader or the viewer will understand it better.

Wed 10 Mar 14:00-15:30

Title	Oral Session
Presenter	Amir Kirsh

2021

Typical Type Typos

AMIR KIRSH[1]

[1] *Academic College of Tel-Aviv-Yaffo, Tel-Aviv, Israel*

Use of proper types is crucial for both performance and correctness. The talk would cover bugs related to type correctness as well as code that may work but hide severe performance issues due to implicit casting, creation of unnecessary temporaries or just simple bad coding. We will walk through abruptly sliced objects, surprisingly dandling pointers, silent temporaries, undefined behavior and more. The typical type typos presented in this talk are quite common, you may find yourself browsing your code right after this session looking for them and meeting them face to face in your code. The talk assumes prior knowledge of rvalue reference and smart pointers.

Wed 10 Mar 14:00-15:30

Title	Oral Session
Presenter	Arno Schoedl

2021

The C++ rvalue lifetime disaster

ARNO SCHOEDL[1], SEBASTIAN THEOPHIL[1]

[1] *Think Cell, Berlin, Germany*

Rvalue references have been with us since C++11. They have originally been introduced to make moving objects more efficient: the object an rvalue reference references is assumed to go out of scope soon and thus may have its resources scavenged without harm. The C++ standard library, for example std::cref or std::ranges, makes use of yet another aspect of rvalue references: since they go out of scope soon, it is assumed unsafe to hold on to them beyond the scope of the current function, while lvalue references are considered safe. We, too, found this assumption to be very useful for smart memory management, in particular in generic code. Unfortunately, the C++ language itself violates this assumption in at least two places. First, rvalues bind to const&. This means that innocent-looking functions taking a parameter by const& and passing it through in some way silently convert rvalues to lvalue references, hiding any lifetime limitation of the rvalues. std::min/max are two such examples. Worse still, every accessor member function returning a const& to a member suffers from this problem. Second, temporary lifetime extension is meant to make binding a temporary to a reference safe by extending the lifetime of the temporary. But this only works as long as the temporary is still a prvalue. If the temporary has been passed through a function, even it has been correctly passed through by rvalue reference, lifetime extension will no longer be invoked and we get a dangling reference. These problems are not merely theoretical. We have had hard-to-find memory corruption in our code because of these problems. In this talk, I will describe the problems in detail, present our library-only approach to mitigate the problems, and finally, make an impossible-to-ever-get-into-the-standard proposal of how to put things right.

Wed 10 Mar 14:00-15:30

Title	Oral Session
Presenter	Luca Minudel

2021

How technical debt can kill your business. How F1 teams crack technical debt.

LUCA MINUDEL[1], **PAOLO POLCE**[2]

[1] SmHarter.com, London, United Kingdom [2] SimulWorks.com, London, United Kingdom

How technical debt can kill your business. How F1 teams crack technical debt. Abstract: Formula One is a multibillion-dollar business, an entertainment, and the pinnacle of the motorsport competition. F1 teams face incredible pressure to deliver. Nonetheless, at every race they manage to deliver and improve at the same time, cracking the technical debt and pursuing technical excellence. Whereas many organisations instead struggle to find the time to do the same under the pressure of their delivery commitments. This session presents 4 real-life scenarios and for each one, an epic-fail story detrimental to the business and a success story from an F1 team, with lessons learned.

Wed 10 Mar 14:00-15:30

Title	Oral Session
Presenter	Peter Sommerlad

2021

Safer C++: MISRA-C++:202x rules and beyond

PETER SOMMERLAD[1]

[1] Better Software: Consulting, Training, Reviews, Wollerau, Switzerland

C++ is a language of choice for implementing software for safety critical or modern embedded systems, however, since its inheritance of many C features and low-level and performance focus it allows for problematic code that still compiles. Not only the risk of incorporating _undefined behaviour_ and non-portability of _implementation-defined behaviour_ can cause safety risks, but also developers misunderstanding the underlying rules of the language. Limiting C++ to a safer core language is the goal of many guidelines in this talk we show rules of a safer subset of C++ for the automotive industry by MISRA-C++:202x. Expect well-known stuff and surprising aspects to be addressed by such rules and what you will get as warnings from the corresponding static analysis tools. However, we will also look at safer C++ design beyond MISRA-C++ rules, because such design issues usually cannot be checked by analysis tools, for example, the use of strong typing and mechanisms that ease following some of the rules, such as the use of sized integeger types.

Wed 10 Mar 14:00-15:30

Title	Oral Session

Wed 10 Mar 16:00-17:30

Title	Oral Session
Presenter	Clare Macrae

2021

Refactoring Superpowers: make your IDE do your work, faster and more safely

CLARE MACRAE[1]

[1] Clare Macrae Consulting Ltd, CAMBRIDGE, United Kingdom

You've got to make a change, and the tests are passing, but you're struggling to get the code to do what you need. You think you can see a way... Maybe the code won't compile for half an hour whilst you bend it to your will... And maybe your code reviewers won't complain about the size of the change, taking them hours to review? And if you're lucky and concentrate very hard, it will be OK. Won't it? As Kent Beck says, "Make the change easy (warning: this may be hard), then make the easy change." This talk will show you techniques to be kind to yourself - and your team - by making seemingly complex edits in small, safe steps, with your IDE doing much of the heavy lifting. You'll be less tired at the end, and confident that the behaviour is unchanged. And users get the feature sooner - win, win!

Wed 10 Mar 16:00-17:30

Title	Oral Session
Presenter	Hendrik Niemeyer

2021

A Practical Introduction to C++20's Modules

HENDRIK NIEMEYER[1]

[1] ROSEN Technology and Research Center GmbH, Lingen (Ems), Germany

Modularity is a long known principle for creating maintainable, changeable and durable software. In contrast to other programming languages, C++ did not have its own module system and used header files to structure code. But this has changed with C++20 which introduced modules as one of its bigger new features. In this talk I will discuss the theoretical side of modules and their advantages over header files (e.g. faster compile times, invisibility of macros, preprocessor directives and nonexported entities, fewer name collisions and more) but the main focus will be the practical usage of modules today. I will demonstrate how modules can be used today with the three major compilers and how to restructure your projects using modules. Also the support of build systems for modules will be discussed.

Wed 10 Mar 16:00-17:30

Title	Oral Session
Presenter	Kris Jusiak

2021

Future of testing with C++20

KRIS JUSIAK[1]

[1] Quantlab Financial, LLC, Denver, The United States of America

Testing in C++ is not easy, it requires a lot of boilerplate code, macro usage and/or understanding of complicated testing frameworks. But it doesn't have to be like that. C++20 will enable us to reinvent the way we write tests. If taking a glance into the future of testing in C++ peaks your interest this session is for you. In this case study we will address the difficulty of testing with C++ by implementing a new, fully functional, macro-free testing framework [1] from scratch with modern C++20 features. The main goal will be to leverage modern C++ in order to make the following snippet compile and run: 1. int main() { 2. "hello world"_test = [] { // Running "hello world"... 3. expect(12_i == fib(7)); // hello_world. cpp:3:FAILED [12 == 13] 4. }; // tests: 1 | 1 failed 5. } // asserts: 1 | 0 passed | 1 failed

The session will also focus on how to design modern testing facilities such as: * Sub/sections * Parameterized tests * Behaviour Driven Development (BDD) At the end of this session the audience will have a better understanding of C++20 features such as: * New additions to lambdas * User Defined Literals (UDL) * Concepts * Source Location As well as how and where to apply them. Additionally, attendees will get familiar with a new, expressive way of testing with modern C++. Let's get ready to test all the things at ACCU 2020 and follow the Beyonce rule - "If you liked it then you should put a test on it". [1]: https://github.com/boost-experimental/ut

Wed 10 Mar 16:00-17:30

Title	Oral Session
Presenter	Andrew Sutton

2021

Reflection: Compile-time Introspection of C++

ANDREW SUTTON[1]

[1] Lock3 Software, LLC, Kent, Ohio, The United States of America

Static reflection is a forthcoming feature in the C++ programming that promises powerful new features for compile-time introspection of user source code. This feature, supported by increasingly capable constexpr facilities, will potentially be the Next Big Thing for C++23. This talk will cover the core concepts of reflection and reification and discuss several methods of providing language support for those features, including the approach taken by current Reflection TS and alternative approaches that are currently being considered. The remainder of the talk will present examples of how reflection can be used to augment traditional generic programming techniques to develop even more general algorithms based on introspection of class properties.

Wed 10 Mar 16:00-17:30

Title	Oral Session
Presenter	Frances Buontempo
Presenter	Steve Love

2021

You can't test this? Hammertime!

FRANCES BUONTEMPO[1], **STEVE LOVE**[2]

[1] BCL, Bradford, United Kingdom [2] Arventech, Bradford, United Kingdom

How many times do people claim testing their code is difficult, if not impossible? Or have hundreds of tests, most of which fail on a regular basis? As data science becomes more prevalent, we often see few useful tests along with the code. Often the "science" relies on data, and the tests end up pulling in entire datasets, run for ages and only state "fail" when they get to the end. You could hit the code with a hammer until the tests pass, ignore the tests or change the asserts, but does that give you confidence to deploy your solution? Even when the code uses randomness, you can test this. We'll investigate some typical problems in machine learning, data science and programming in general, suggesting varied approaches to testing in order to increase confidence that your code Does The Right Thing (TM).

Thu 11 Mar 09:00-10:30

Title	Invited Slot
Presenter	Kevlin Henney
Guest Presenter	Kevlin Henney

2021

Keynote: It Depends...

KEVLIN HENNEY[1]

[1] *Kevlin Henney, Bristol, United Kingdom*

When you start something new, you soak up guidelines — rules that guide, lines to not cross — to find your way. You make progress by following the always do this and avoiding the never do that. With experience comes the realisation that there are exceptions to these rules. With expertise comes the realisation that these are not exceptions, but highly context-dependent rules. The always do this and never do that you've relied on give way to it depends. Former certainties dissolve into a sea of possibilities. Context and dependency on context are hard to see, but they are critical to thinking and to code. The creation of software is an exercise in knowledge acquisition and retention, and there are limits to what we know and what we can know, and limits to what can be stated and tested, but seeing software as knowledge structure has important consequences. All too often, assumptions about code dependencies and execution, remain unstated and unexplored, hidden behind not-quiteright mental models and weak formalisms for reasoning. Ask any developer to draw up the dependencies in their current system and, after a few boxes and lines, they'll likely run dry, having barely sketched the tip of the tip of the iceberg. Most dependencies are out of sight and out of mind. Mentions of technology and third-party code are vague (versions and vendors completely overlooked), risk implications are unacknowledged (from left-pad to Heartbleed), the roles of developers and organisations and the real world are considered a mildly inconvenient abstraction. This is software development. This is water. This is dependency.

Thu 11 Mar 11:00-12:30

Title	Oral Session
Presenter	Andy Balaam

2021

Interesting Characters

ANDY BALAAM[1]

[1] *OpenMarket, London, United Kingdom*

When I started learning how Unicode works, I knew it would be useful, but I never expected it to be so much fun. In this talk I will try to share my excitement! We will tour some of my favourite characters and use them to help us understand how various character sets and encodings work, covering Unicode in most detail, but also mentioning the Latin-n family, GB18030 and even EBCDIC. By the end of this session you will have strong opinions about how UTF-16 ruined everything, how UTF-8 is beautiful nonetheless, and maybe you'll even have a favourite Unicode character yourself. Personally I am a big fan of U+FE18.

Thu 11 Mar 11:00-12:30

Title	Oral Session
Presenter	Seb Rose

2021

Contrasting test automation and BDD: an "interactions over tools" perspective

SEB ROSE[1]

[1] *SmartBear, Edinburgh, United Kingdom*

Test automation and BDD are related, but they are not the same. To get the most out of each of them, we need to understand the separate challenges that they address before getting engrossed in the tools that have been created to facilitate their adoption. And those challenges are rooted in the interactions between the different disciplines involved in software specification and delivery.

In this session we'll explore what test automation and BDD are - and how they separately contribute to successful inter-disciplinary agile delivery. We'll also spend some time describing how they're different, and look at several typical examples of what can go wrong when BDD and test automation get confused.

*** This session includes some short exercises for participants, using Zoom break-out rooms. It is helpful if you could have a working microphone and (optionally) camera ***

Thu 11 Mar 11:00-12:30

Title	Oral Session
Presenter	Sebastian Theophil

2021

Windows, macOS and the Web: Lessons from cross-platform development at think-cell

SEBASTIAN THEOPHIL[1]

[1] *think-cell, Berlin, Germany*

For twelve years, think-cell had been a Windows-only software company and our codebase of approximately 700k lines of code had accumulated many unintentional platform dependencies. Six years ago, we decided to port our application to the Mac. This change has affected every part of our development process: the project organization, build system and the way we program in C++ today. The commonly used cross-platform libraries such as Qt and boost were good tools to build on, but by themselves were not enough. For many concepts, such as mutexes, semaphores or shared memory, they only offer a common interface to platform-specific objects with very different semantics and lifetimes. We wanted light-weight, platform-independent C++ abstractions with identical semantics for rendering, internationalization, file I/O, mouse event handling, RPC calls and error reporting. Developing these was challenging, firstly, because we had to define which semantics our application needed and, secondly, we had to implement them on each platform. This was not an easy process but I would argue it has improved the quality of our code very much. By now, we have moved on to the next challenge and have started to move some functionality to web applications. We wanted to reuse our existing code-base of course, and that meant writing web applications in expressive, typesafe C++. Definitely an advantage in our book! We have built our web applications using emscripten, but thanks to a student intern, we generate type-safe C++ bindings, beyond those provided by emscripten, from any typescript interface definition. In my talk, I will give you an overview of the C++ abstractions we have implemented, focusing on the cross-platform problem areas where common semantics were hard to define due to limitations of either one of the operating systems, and of course I will show you our tools that let us write web application in C++.

Thu 11 Mar 11:00-12:30

Title	Oral Session
Presenter	Tina Ulbrich

2021

Generic Programming without (writing your own) Templates

TINA ULBRICH[1]

[1] *ROSEN Technology and Research Center GmbH, Lingen (Ems), Germany*

Generic programming is a technique where functions and data structures are defined as general as possible. The goal is that they work with different data types and therefore are reusable. In C++ generic functions and data structures are typically realizes by using templates. Templates are a great tool for generic programming but they come with their own set of challenges. They can be hard to read/write and influence compile times negatively. And sometimes they are even too generic. C++20's concepts can help with that but I want to show how you can be generic in your code without writing your own templates. To do that, we will explore some C++17 and C++20 features from the standard library, like std::span, std::variant and std::any. I will explain what they are, where they are useful and how to use them. I will also show some features you can find in other libraries, e.g. GSL.

Thu 11 Mar 11:00-12:30

Title	Oral Session
Presenter	Tristan Brindle

2021

An Overview of Standard Ranges

TRISTAN BRINDLE[1]

[1] *Tristan Brindle, London, United Kingdom*

The ranges revolution is nearly upon us! C++20 will include concept-enabled, range-based versions of all the standard algorithms you know and love, as well as new "views" which can transform the way you write code. In this talk we'll offer an overview of the ranges features currently in the C++20 draft, with examples of how you can use them to reduce verbosity, avoid bugs and improve the correctness of your code, and in some cases get better performance. We'll also cover the currently-available ranges implementations that you can use today, without having to wait for the next version of the standard. If you've heard the buzz around ranges and are wondering what they'll bring and how they'll benefit your code-base, then this is the talk for you.

Thu 11 Mar 12:45

Title	Oral Session
Presenter	Chris Croft-White

Time Travel Debugging - it's time to Debug Different

CHRIS CROFT-WHITE - ENGINEER

Time Travel Debugging simplifies debugging by letting developers freely step backwards, as well as forwards, through a program's execution.

In this practical lunch and learn session we'll demonstrate benefits of Time Travel debugging and show how it delivers a radically simplified workflow for debugging, compared to traditional methods of debugging forwards.

Our Engineer will walk through how developers can simplify fixing bugs by:

Deterministically capturing and replaying a program's behavior

Time traveling backward (and forward) in the code execution to the origin of the failure

Reducing the amount of repetitive guesswork and time consuming restarts and stops normally involved in debugging

The demonstration will feature UDB, Time Travel Debugger and the open source debugger, RR.

Thu 11 Mar 14:00-15:30

Title	Oral Session
Presenter	Chris Oldwood

2021

PowerShell for the Curious

CHRIS OLDWOOD[1]

[1] *Freelance, Godmanchester, United Kingdom*

PowerShell was unleashed on Windows developers and administrators back in 2006 in an effort to fill the long standing Windows CLI void. Fast forward 10 years and the shell-cum-scripting language can now also be found on macOS and Linux as part of Microsoft's new found love for Open Source and cross-platform tooling. Unlike traditional shells which traffic in lines of text, PowerShell leverages .Net Core and allows whole objects, generated by its own 'cmdlets', to be passed through pipelines alongside the simple lines of text generated by the vast catalogue of existing command line tools we all know and love. As a consequence this shell / scripting language hybridization means that administrators and developers can solve problems using whichever paradigm they are most comfortable with. Throw in support for remoting, modules, a package manager, the native .Net Core libraries, etc. and you have a mature ecosystem for writing everything from cryptic one-liners to fullblown applications. If PowerShell has entered your conscious incompetence and you're curious about what it might offer, this talk introduces you to the basic concepts and aims to give you enough of a grounding to help you explore further.

Thu 11 Mar 14:00-15:30

Title	Oral Session

Thu 11 Mar 14:00-15:30

Title	Oral Session
Presenter	Alan Griffiths

2021

What use is a confined user shell?

ALAN GRIFFITHS[1]

[1] *Canonical, United Kingdom*

A user shell is the way in which a user interacts with a computer system. It is the way in which input from keyboards, mice, touchscreens etc reach the system and applications and the way that output reaches the screen. A shell is responsible for launching applications, routing input to the focussed application and compositing the output from the visible applications onto the display. Confinement is a way of restricting the capabilities of a program to specific devices, parts of the filesystem and other features of the system. An "unconfined" program can access anything that the user has permissions to do. So, for example, it could read any of the user's files and copy them to the internet. The use of confinement is of increasing importance in computing as the basis for trust between the developer of a program and its user becomes increasingly tenuous. Drawing from experience with adapting graphical shells and other applications to "confined" execution we examine what is needed to securely run untrusted applications on a computer.

Thu 11 Mar 14:00-15:30

Title	Oral Session

Thu 11 Mar 14:00-15:30

Title	Oral Session

Thu 11 Mar 14:00-14:20

Presenter	Victor Ciura

2021

C++ UNIverse

VICTOR CIURA[1]

[1] *CAPHYON, Craiova, Romania*

Performance has always been the goal for C++ and that can frequently come in conflict with teachability. Since I was a student, twenty years ago, until today C++ has been a staple diet in universities across the globe. But "C++ as a first language"... really? There is a lot of room for us to make C++ more teachable and improve the quality of C++ teaching in UNI, so long as we're not talking about CS1. First, students have to get over the hurdle of being algorithmic thinkers and then we can give them a language that has these sharp edges. Is this a lost cause? I think not. Modern C++ is simpler and safer and we have numerous opportunities to make it more teachable at the same time. "The king is dead, long live the king!"

Thu 11 Mar 14:00-14:20

Presenter	Piotr Gaczkowski

2021

Building portable C++ packages: the Curse of Abundance

PIOTR GACZKOWSKI[1]

[1] *Meraki Acoustic, Gdansk, Poland*

Some people believe that freedom of choice is the best thing we can get. If you've built a few C++ projects in your lifetime you may have experienced the freedom of choice: Makefile, autotools, shell scripts, Waf, Bazel, Maven, QMake, CMake, ...! This buffet is much more than any single person can eat! When building Songcorder, a vinyl-to-digital converter, I had to integrate libraries built with a combination of Makefiles, Waf, CMake, and Conan. And to make them work on Linux, Mac, and Windows. With a nice CI/CD pipeline. I came up with a fusion dish that neither looks or tastes very good. But at least it satisfies the hunger. I'll share with you some of my insights into the current landscape of build tools and what I learned on this journey.

Thu 11 Mar 14:20-14:40

Presenter	Ahto Truu

2021

Programming as a sport -- what do you mean?

AHTO TRUU[1]

[1] *Guardtime, Tallinn/Tartu, Estonia*

Humans tend to be competitive. With that in mind, it is perhaps not surprising that anything can be turned into sports, including intellectual activities like programming. This short presentation intends to give an overview of the world of competitive programming and also reflect a bit on its relationship to software engineering. If there is enough interest, we may follow up with a practice session after the hours!

Thu 11 Mar 14:20-14:40

Presenter	Adrian Ostrowski

2021

Building portable C++ packages: the bliss of unification

ADRIAN OSTROWSKI[1]

[1] *Intel, Gdansk, Poland*

There are many ways to build and package your C++ code, each with a different approach to supporting multiple operating systems and toolchains. Integrating third-party components into your software often means having to deal with even more build systems. All of that without even mentioning topics like cross-compilation. How to grasp and handle all of this complexity? In this talk, you will be presented with a portable solution to building and packaging your code using CMake. You'll discover how to easily create and use toolchains for different platforms. Lastly, you'll learn how to package your code and easily leverage pre-built packages using Conan.

Thu 11 Mar 14:45-15:05

Presenter	Amir Kirsh

2021

The Point Challenge - returning different types for the same operation

AMIR KIRSH[1]

[1] *Academic College of Tel-Aviv-Yaffo, Tel-Aviv, Israel*

Types are important as a tool for enforcing program correctness. In this session we would discuss types, specifically - Point. There is no real logic in adding up two points (result is meaningless) - so we probably may not implement operator+ for Point... Unless we want to calculate an average location, then summing 2 points and dividing by two should be supported, preferably treating the result as a real Point only after the division by 2. We would discuss this problem and see how we can implement methods that return different types for (almost) the same operation and enforcing compile time type correctness. Discussion would go through actual code and usage of cool C++ features such as if constexpr, advnaced template techniques and more.

Thu 11 Mar 14:45-15:05

Presenter	Richard Wallman

2021

Testing your tests with code coverage

RICHARD WALLMAN[1]

[1] *Civico Ltd, Birmingham, United Kingdom*

Everyone is writing tests for their code, right? Having tests is a good start, but unless you're testing every single line of code, there's still chances for "fun" times with bugs. Having an incomplete set of tests can provide a false sense of security, but manually checking every possible execution path is tedious and error-prone. Thankfully, there are tools which, when used in conjuction with our test suite, can highlight code that is never executed - the gaps in our test cases. Armed with this information, we can add to our existing test suites to cover these (hopefully edge) cases. This talk will focus on the GNU Compiler Collection, but other compilers (such as Clang) also include similar tools. This talk is about the process, not the tools.

Thu 11 Mar 14:45-15:05

Presenter	Natalia Pryntsova

2021

Services evolution: required is forever

NATALIA PRYNTSOVA[1]

[1] Bloomberg LP, NY, The United States of America

With more and more systems moving to a microservices architecture, we often find ourselves designing new integration endpoints and scrutinizing message schemas. As we design new service APIs, one thing is certain – at some point, it will need to change, preferably without breaking every service around it. And this is where backward compatibility matters.

This talk is about schema versioning and different options to consider when designing services. Binary serialization or JSON? Shared schemas or schema-less? We will start with a brief overview of backward and forward compatibility and why both are important, but tricky to achieve in practice. Next, we will dive into details of JSON and binary serialization issues. For binary serialization, we will use Apache Avro and Google Protobuf as examples and we will explore implementation details of integer packing and the usage of field IDs. Finally, we will summarize how encoders drive compatibility and, eventually, service evolution.

Thu 11 Mar 15:10-15:30

Presenter	Jim Hague

2021

Handling large volumes of immutable structured data with ClickHouse

JIM HAGUE[1]

[1] Sinodun Internet Technologies Ltd, Oxford, United Kingdom

Are you on the receiving end of a firehose of immutable structured data, which you want to analyse using SQL queries? Have you come across ClickHouse before? ClickHouse is an open-source columnoriented database management system with SQL querying written in C++. For the past few years I've been using it at the heart of an analytics application handling 12 billion records of DNS transactions daily on a cluster of 4 servers. This talk will give an introduction to ClickHouse and demonstrate what you can do with it and a large lump of data on a humble laptop.

Thu 11 Mar 15:10-15:30

Presenter	Lotte Steenbrink

2021

Tools that spark joy: lessons learned from the Rust ecosystem that can be adopted elsewhere

LOTTE STEENBRINK[1]

[1] Ferrous Systems, Berlin, Germany

Tooling is an essential part of creating solid software: it helps us find errors, build what we want and understand what we've built. With a compiler that's eager to help, accessible documentation and modern dependency management, Rust consistently scores highest in developer surveys when it comes to development tools. Working with Rust in 2020, I learned just how much its ecosystem lives up to this reputation, and wished I'd had a similar experience in C and C++ projects. However, switching to Rust shouldn't be the only way to have this aha moment. Instead, we can examine Rust's tooling to learn how it manages to provide you with exactly the help you need - exactly when you need it, and apply this knowledge to the helpers we build for our own ecosystem. This talk examines the culture, technical insights and resulting design decisions that shaped how the Rust community approaches tooling. We'll be looking at core infrastructure like rustc and learn how their best practices organically influenced third-party efforts such as knurling-rs. We'll wak through what Rust's friendly tooling looks like in practice, and which lessons learned from building it can be applied to improve development workflows in other languages too.

Thu 11 Mar 15:10-15:30

Presenter	Seb Rose

2021

Example mapping: a structured, collaborative discovery technique

SEB ROSE[1]

[1] SmartBear, Edinburgh, United Kingdom

Is your team struggling with unproductive meetings and workshops? Are you unsatisfied with how your team comes together to refine requirements and specify solutions? Have you heard about example mapping and want to know more? Specifying and delivering software is a process of discovery. No team has ever delivered a valuable product without discovering many things during the development process, but many teams struggle to get good at discovery. Matt Wynne created a technique called example mapping that has helped thousands of teams around the world use examples to reach a shared understanding of the problems that need solved. As a consequence there are fewer misunderstandings, fewer disagreements, and a smoother flow of value delivery. This session will teach you what example mapping is and why it works. Through a series of practical demonstrations, we will explore the essential actions needed to prepare for, facilitate, and derive value from example mapping. This knowledge will then make it simple for you and your teams to adopt example mapping successfully (using the many excellent online teaching resources available) and adapt it to your specific context.

Thu 11 Mar 16:00-17:30

Title	Oral Session
Presenter	Greg Law
Presenter	Dewang Li

Reconfirmed

Modern Linux C++ debugging tools - under the covers

GREG LAW[1], DEWANG LI[2]

[1] Undo, Cambridge, United Kingdom [2] Synopsys Software Integrity Group, Mountain View, The United States of America

An overview of how some of the seemingly-magical modern Linux C++ tools actually work so that you can get the most from them. C++ is a language and ecosystem that is unashamedly close to the metal, and to be an expert practitioner an understanding of compiler and OS fundamentals is essential, and this includes debugging and profiling tools. The last decade has seen a 'cambrian explosion' in tooling: Valgrind, perf, Address Sanitizer, rr, Live Recorder, Coverity and cppcheck have either arrived or become mainstream and even good old GDB has come a long way. Greg gives an overview of how these amazing/magical tools are implemented often exploiting a combination of compiler, OS and CPU features. Contains details on ptrace, DWARF debug info, how static analyzers work, record and replay systems - so that you can select the right tool for the job and then get the most out of it.

Thu 11 Mar 16:00-17:30

Title	Oral Session
Presenter	Bob Steagall

Reconfirmed

The Business Value of a Good API

BOB STEAGALL[1]

[1] KEWB Computing, Gaithersburg, MD, The United States of America

As programmers, we use APIs every day, whether it is at the library level, the subsystem level, or the individual component level. And yet, using many existing APIs is often an unsatisfying experience, for any number of reasons: poor documentation, confusing component interfaces, muddled abstractions, broken/missing functionality, etc. So how can we distinguish between a good API and a bad API? More importantly, how can we make the argument to our managers that good APIs, whether obtained off-the-shelf or built in-house, are a prudent investment? This talk seeks to discuss these questions and perhaps provide some answers. We'll look at specific criteria for evaluating the goodness and/or badness of an API. We'll cover the concepts of technical debt and software capital, and define a relationship between them and API goodness/badness. We'll also discuss a simple iterative process for building APIs which can help avoid technical debt and increase software capital.

Finally, we'll cover some recommendations for doing evaluations, using the process in day-to-day work, and convincing management of your wisdom.

Thu 11 Mar 16:00-17:30

Title	Oral Session
Presenter	Zhihao Yuan

2021

Thinking in Immediate: ImGUI

ZHIHAO YUAN[1]

[1] SimpleRose Inc, St. Louis, The United States of America

When programming graphical user interfaces, sometimes we might think, "Ah, I wish to have an event listener that triggers when this data member changes!" Before laughing at yourself, adding getters and setters everywhere so that you can emit a signal whenever you want, I want to tell you, your naive thought is actually the right way of thinking about that program. If you have data, the GUI should follow the data. Data change GUI change. Two widgets use that data, two widgets change at the same time. That is, Immediate Mode GUI. This talk will introduce immediate mode GUI programming with *pyimgui*, a Python library that pushes the elegance of the Dear ImGUI library in C++ to a new boundary. This time, let's think in immediate, think functional, express your program with no callbacks, and bring back the joy of programming.

Thu 11 Mar 16:00-17:30

Title	Oral Session
Presenter	Erika Sweet

2021

Cross-Platform Pitfalls and How to Avoid Them

ERIKA SWEET[1]

[1] *Microsoft, Redmond, Washington, The United States of America*

C++ cross-platform development is difficult. These difficulties are compounded by the fractured solution space, where every project seems to use a different combination of build systems, package managers, and diagnostic tools to address shared challenges. Join us for a discussion and demo of C++ cross-platform development centered on common pitfalls and widely adopted tooling.

Learn how to leverage CMake and its new CMakePresets.json to seamlessly build across operating systems and platforms. Untangle your dependencies with tools like vcpkg and Conan to avoid inconsistencies between system package managers. Debug your projects across multiple platforms with remote debugging. We'll also explore how CMakePresets.json is supported on the CMake command line, in Visual Studio, and in Visual Studio Code.

Thu 11 Mar 16:00-17:30

Title	Oral Session
Presenter	Alisdair Meredith

Reconfirmed

Frictionless Allocators

ALISDAIR MEREDITH[1]

[1] *Bloomberg LP, New York, The United States of America*

The benefits of users taking control over their own memory allocation strategy have been demonstrated many times at previous ACCU conferences. However, despite these benefits and ongoing support in the C++ Standard, allocator aware software has not yet become widespread throughout the C++ community. One of the main sources of resistance is perceived complexity when providing allocator support in our libraries, which is the prerequisite for empowering users to make choices optimal to their circumstances. This talk will tackle that complexity head-on, seeking to remove the friction between library support and user. First, we will examine the sources of friction when writing allocator awre code in C++ today. Then, we will then suggest a small selection of potential language extensions that would permit much cleaner expression of the same designs deployed today. We aim to take the friction out of the system! This talk provides an early preview of several language proposals we hope to send to the ISO C++ committee for C++23 and beyond; it will touch on lessons learned from an early prototype implementation; and we will discuss how to judge when the proposal and experience with it are mature enough to take up valuable committee time to move forward!

Fri 12 Mar 09:00-10:30

Title	Invited Slot
Guest Presenter	Patricia Aas

Confirmed

Keynote: Who are they, and what do they want?

PATRICIA AAS[1]

[1] *Patricia Aas, Oslo, Norway*

Over the last two decades the security posture of operating systems, hardware vendors and compilers have dramatically improved. Interestingly, during that same period, we have seen a rising level of "professionalism" on the part of those that do binary exploitation. What does that mean for us as programmers? Where are we vulnerable? And maybe most of all: who are our adversaries, what are they trying to achieve and what are their capabilities?"

Fri 12 Mar 11:00-12:30

Title	Oral Session
Presenter	Victor Ciura

2021

AddressSanitizer on Windows

VICTOR CIURA[1]

[1] *CAPHYON, Craiova, Romania*

Clang-tidy is the go-to assistant for most C++ programmers looking to improve their code, whether to modernize it or to find hidden bugs with its built-in checks. Static analysis is great, but you also get tons of false positives. Now that you're hooked on smart tools, you have to try dynamic/runtime analysis. After years of improvements and successes for Clang and GCC users, LLVM AddressSanitizer (ASan) is finally available on Windows, in the latest Visual Studio 2019 versions. Let's find out how this experience is for MSVC projects. We'll see how AddressSanitizer works behind the scenes (compiler and ASan runtime) and analyze the instrumentation impact, both in perf and memory footprint. We'll examine a handful of examples diagnosed by ASan and see how easy it is to read memory snapshots in Visual Studio, to pinpoint the failure. Want to unleash the memory vulnerability beast? Put your test units on steroids, by spinning fuzzing jobs with ASan in Azure, leveraging the power of the Cloud from the comfort of your Visual Studio IDE.

Fri 12 Mar 11:00-12:30

Title	Oral Session
Presenter	Charles Weir

2021

Playing with Security

CHARLES WEIR[1]

[1] *Lancaster Univesity, Lancaster, United Kingdom*

Software security is scary, right? Well, no, it doesn't have to be. In this workshop we'll learn, by doing, how to understand software security and the decisions we need to make. In the Agile Security Game, we'll work in teams, prioritising security improvements to an app-based payment application and its infrastructure. We'll learn from the discussions and from the impact of security events following our decisions. Not only is the game fun, it also provides an excellent way of helping colleagues back at work to discuss and approach the issues of software security. As a participant, you'll receive instructions for how to set up similar game workshops yourself and we'll discuss how best to introduce them in different team environments. If you were at Angela Sasse's ACCU 2019 keynote (https://www.youtube.com/watch?v=tBmF7ofKoYQ), this workshop picks up on many of the issues she raised. The workshop does not depend on having seen Angela's keynote, however, and anyone can attend.

Fri 12 Mar 11:00-12:30

Title	Oral Session
Presenter	Eoin Woods

2021

API Vulnerabilties and What to Do About Them

EOIN WOODS[1]

[1] *Endava, London, United Kingdom*

Thanks to the increasingly dangerous threat landscape, a large number of high profile security breaches, and the tireless work of organisations like OWASP, security is finally becoming a high priority topic in many software development projects. For many years OWASP have provided simple, practical guidance on security to software developers, their best known output probably being their "Top 10" lists of vulnerabilties for webapps and mobile development. However in recent years the explosion in popularity of application APIs has opened up another dangerous attack vector in many systems. In response, OWASP have recently developed their "API Security Top 10" list to provide similar guidance for APIs.

In this talk we will review the current security landscape, particularly as it relates to API-based applications, and explore the API Security Top 10 vulnerabilities in order to understand the top security threats to our APIs, which ones we might have missed in our systems, and what practical mitigations we can use to address them when we get back to work after the conference.

Some of this (such as logging and monitoring) will probably be familiar to those who who are already aware of the webapp Top 10, but is likely to bring a different perspective to it, while other parts (such as payload related problems) is likely to be new.

Fri 12 Mar 11:00-12:30

Title	Oral Session
Presenter	Arno Schoedl

2021

From Iterators To Ranges — The Upcoming Evolution Of the Standard Library

SEBASTIAN THEOPHIL [1], **ARNO SCHOEDL**[2]

[1] *think-cell, Berlin, Germany* [2] *Think Cell, Berlin, Germany*

Pairs of iterators are ubiquitous throughout the C++ library. It is generally accepted that combining such a pair into a single entity usually termed Range delivers more concise and readable code. Defining the precise semantics of such Range concept proves surprisingly tricky, however. Theoretical considerations conflict with practical ones. Some design goals are mutually incompatible altogether.

Fri 12 Mar 11:00-12:30

Title	Oral Session
Presenter	Patrick Martin

2021

Hideous Mathematics for the software engineer

PATRICK M. MARTIN[1]

[1] *Bloomberg LP, London, United Kingdom*

= Why the word "hideous"? This is not a comment on mathematics per se, but on the consequences of having an inappropriate model of the work we do. = The proposition Programming is an inherently mathematical discipline. The supporting raw electronics and computing components only follow simple rules, yet have the capacity for the most complex outcomes. For these systems it is clear that having a good mathematical intuition of the properties of such components is essential for any in depth work. I propose that this paradigm remains relevant as we move up to higher abstractions in the computing stack towards the realm of the human elements and the projects they conceive. I will visit a number of models, which either have heuristics or suggest processes, and discuss if, how and when they benefit us when applied to programmers and products. Finally, if there is a suggestion for the optimum course of action when applying a model, this implies a penalty for applying a less apt model, or applying the better model, but skipping more appropriate processes. Less tactfully: what happens when we get things wrong? Given this will be a topic in which everyone in the room will be an expert, expect some audience participation!

Fri 12 Mar 14:00-15:30

Title	Oral Session
Presenter	Dom Davis

2021

Hi, I'm Dom, and I Have Depression!

DOM DAVIS[1]

[1] *Tech Marionette, Norwich, United Kingdom*

Which for a session title at a tech conference isn't what you expect. And this isn't what you expect from an abstract. The session probably isn't what you expect either. Of course, to look at me, you wouldn't realise half the problems I deal with, because a lot of the time I'm hiding behind the logic of code, and the shield of electronic communication. Turns out you wouldn't realise it with most people. We're fun like that. In this session we're going to look at neurodiversity. What it can mean for your team. What it can mean for you; regardless of if you classify yourself as neurodiverse or not. You can join in, or you can sit and listen. It's up to you. The idea is to have fun, share ideas, and learn.

Fri 12 Mar 14:00-15:30

Title	Oral Session
Presenter	Rob Richardson

2021

JavaScript the Grumpy Parts

ROB RICHARDSON[1]

[1] *@rob_rich, Gilbert, AZ, The United States of America*

We love JavaScript, but we must admit: it's weird. Why does `this` behave as it does? How does variable scope work? Why do we have such comical behavior when comparing mixed types? Let's pull back the covers of these scenarios, and learn how it truly works. You may find a new reason to fall in love with JavaScript.

Fri 12 Mar 14:00-15:30

Title	Oral Session
Presenter	Ahto Truu

2021

How to build digital signatures from hash functions

AHTO TRUU[1]

[1] *Guardtime, Tallinn/Tartu, Estonia*

In modern societies, more and more paper documents and ink signatures are replaced with their electronic equivalents. Now the ascent of quantum computing threatens to render all current digital signature systems insecure. Hash functions, however, seem to be quite resilient to quantum attacks and thus a promising building block for future cryptographic protocols. The talk will recap the essentials of the existing digital signature systems as well as cryptographic hash functions and then show how the former can be built from the latter. Curiously enough, a digital signature system based on hash functions was one of the first to be invented when the idea of asymmetric cryptography was introduced to the world! In addition to reviewing the most important historical systems, the talk will also cover a brand new one developed over the past few years.

Fri 12 Mar 14:00-15:30

Title	Oral Session
Presenter	Mateusz Pusz

2021

Rethinking the Way We Do Templates in C++

MATEUSZ PUSZ[1]

[1] *Epam Systems | Train IT, Gdansk, Poland*

Template metaprogramming is hard. In case it is hard only for the library implementer then it is not that bad. The problem arises when it also affects the users of this library. This talk is summarizing my experience and thoughts gathered during the implementation of the Physical Units Library for C++. The way we do template metaprogramming now results with inscrutable compile-time errors and really long type names observed while debugging our code. That is mostly caused by aliasing class template specializations of non-trivial metaprogramming interface designs. Compilation times of such code also leave a lot of room for improvement, and the practices we chose to use in the Standard Library are often suboptimal. Taking into account the Rule of Chiel while designing templated code makes a huge difference in the compile times. This talk presents a few simple examples (including the practices from the C++ Standard Library) of achieving the same goal in different ways and provides benchmark results of time needed to compile such source code.

Fri 12 Mar 14:00-15:30

Title	Oral Session
Presenter	Luca Sas

2021

Modern C and what we can learn from it

LUCA SAS[1]

[1] *Creative Assembly, Horsham, United Kingdom*

C is often perceived as an antiquated language that is mostly used for legacy purposes, but many people still prefer coding in C or in a subset of C++ that is very close to C. This is sometimes labeled "Modern C" and the ideas behind it are becoming more popular alongside other paradigms such as Data Oriented Design (DOD), Zero Is Initialization (ZII) and Centralized Memory Management (CMM). In fact, many new systems programming languages, such as Rust, Go and Zig, also embody a lot of similar ideas as Modern C showcasing a high degree of interest in these paradigms. In this talk we will explore how programming looks like with this different approach to C and how it addresses matters such as API design, error handling and resource management as well as what are the benefits and costs of these techniques. By the end we will gain a better understanding of how these ideas can help improve the quality of our code, what new languages have adopted similar concepts and to what extent, and what lessons we can learn from this in order to improve our own existing codebases and styles of coding.

Fri 12 Mar 16:00-17:30

Title	Oral Session
Presenter	Pete Muldoon

0200

Redesigning Legacy Systems - Strategies that work / Lessons learned

PETE MULDOON[1]

[1] *Bloomberg LP, New York, The United States of America*

In this presentation, the focus will not be on code but on what most developers will face at one time or another and that is having to redesign and replace an existing legacy system. Production code written in the "Dark ages" with a large user base is hitting its limits in terms of performance, testability and maintainability. This talk looks at what determines when a product needs a partial or full rewrite. What tools and ingredients are needed before you start and how to get rolling. We will also examine the perils and pitfalls of the various stages in the redesign of a legacy system that can slow down or even derail getting the Product out the door and how to avoid them. Finally we will examine how to navigate the nightmare of rolling this new software out to the existing user base. Although the real world examples draw primarily on systems written in C++, the above can be applied to most any complex system being developed by a Team of developers.

Fri 12 Mar 16:00-17:30

Title	Oral Session
Presenter	Conor Hoekstra

2021

C++ Concepts vs Rust Traits vs Haskell Typeclasses vs Swift Protocols

CONOR HOEKSTRA[1]

[1] *NVIDIA, Toronto, Canada*

C++20 comes with Concepts - one of the four major features of the C++20. This talk will explore a basic introduction to what Concepts are, how to use them and how to write one. The talk will also focus on how they compare to "adjacent" language features such as Rust Traits, Haskell Typeclasses and Swift Protocols. This talk will be a "must see" for programming language enthusiasts.

Fri 12 Mar 16:00-17:30

Title	Oral Session
Presenter	Eoin Woods

2021

API Vulnerabilties and What to Do About Them

EOIN WOODS[1]

[1] *Endava, London, United Kingdom*

Thanks to the increasingly dangerous threat landscape, a large number of high profile security breaches, and the tireless work of organisations like OWASP, security is finally becoming a high priority topic in many software development projects. For many years OWASP have provided simple, practical guidance on security to software developers, their best known output probably being their "Top 10" lists of vulnerabilties for webapps and mobile development. However in recent years the explosion in popularity of application APIs has opened up another dangerous attack vector in many systems. In response, OWASP have recently developed their "API Security Top 10" list to provide similar guidance for APIs.

In this talk we will review the current security landscape, particularly as it relates to API-based applications, and explore the API Security Top 10 vulnerabilities in order to understand the top security threats to our APIs, which ones we might have missed in our systems, and what practical mitigations we can use to address them when we get back to work after the conference.

Some of this (such as logging and monitoring) will probably be familiar to those who who are already aware of the webapp Top 10, but is likely to bring a different perspective to it, while other parts (such as payload related problems) is likely to be new.

Fri 12 Mar 16:00-17:30

Title	Oral Session
Presenter	John Lakos

2021

Lakos'20: The "Dam" Book is Done!

JOHN S. LAKOS[1]

[1] *Bloomberg LP, NYC, The United States of America*

Writing reliable and maintainable C++ software is hard. Designing such software at scale adds a new set of challenges. Large-scale systems require more than just a thorough understanding of the logical design concepts addressed in most popular texts. To be successful on an enterprise scale, developers must also address physical design, a dimension of software engineering that may be unfamiliar even to expert developers. More than two decades in the making, Large-Scale C++, Volume I: Process and Architecture, is finally here! Drawing on his over 30 years of hands-on experience building massive, mission-critical enterprise systems, John Lakos — using select excerpts from this glisteningly new volume — elucidates the essential value of (and several techniques needed for) creating and growing hierarchical reusable software, a.k.a. Software Capital, as the foundation for developing C++ software at virtually unbounded scale.

Fri 12 Mar 16:00-17:30

Title	Oral Session
Presenter	Vittorio Romeo

2021

C++11/14 at scale - what have we learned?

VITTORIO ROMEO[1]

[1] *Bloomberg LP, London, United Kingdom*

Many years have passed since the release of C++11 and C++14. These standards brought many new features and idioms to the C++ language and revitalized its community. Nowadays, with C++20 having one foot out the door, it is important to look back at the experience gained using C++11/14 at scale and re-evaluate their impact. - What have we learned from 8 years of using Modern C++ in production at a large-scale corporation? - What features were the most useful? - Which ones were the most misused? From unexpected benefits/ drawbacks to teachability issues, this talk will discuss the most significant consequences of embracing C++11 and C++14 in a company with thousands of engineers. With some _healthy_ skepticism, commonly used features and idioms will be reassessed to uncover some unexpected pitfalls or qualities.

Sat 13 Mar 09:30-11:00

Title	Oral Session
Code	209
Presenter	Andreas Fertig

2021

C++20 Templates - The next level: Concepts and more

ANDREAS FERTIG[1]

[1] *Unique Code, Stuttgart, Germany*

C++20 is probably the biggest change to the language since ever. In this session, we will look into some changes that templates received with C++20. The biggest change is the introduction of Concepts. We don't stop there. We will also talk about improvements to CTAD and NTTP and smaller improvements like explicit(bool). Of course, we will also look into how templated lambdas work in C++20. By the end of the talk, attendees have learned about the newest C++20 template updates and how to apply them.

Sat 13 Mar 09:30-11:00

Title	Oral Session
Presenter	Anthony Williams

2021

Concurrency in C++20 and beyond

ANTHONY WILLIAMS[1], [2]

[1] *Just Software Solutions Ltd, St Just, United Kingdom* [2] *Engineered Arts Ltd, Falmouth, United Kingdom*

C++20 is set to add new facilities to make writing concurrent code easier. Some of them come from the previously published Concurrency TS, and others are new, but they all make our lives as developers easier. This talk will introduce the new features, and explain how and why we should use them. The evolution of the C++ Concurrency support doesn't stop there though: the committee has a continuous stream of new proposals. This talk will also introduce some of the most important of these, including the new Executor model. These include `std::jthread`, which provides automatic joining of threads, `std::stop_token` for cooperative signalling of shutdown, `std::latch` for notifying when a batch of operations have completed, `std::barrier` for multithreaded loop synchronization, and `std::counting_semaphore` for general, flexible synchronization

Sat 13 Mar 09:30-11:00

Title	Oral Session
Presenter	Jim Hague

2021

Building and organising a multi-platform development code base

JIM HAGUE[1]

[1] *Sinodun Internet Technologies Ltd, Oxford, United Kingdom*

There must be few codebases that don't face some sort of cross-platform or multi-platform challenge. The size of the challenge varies. At its simplest it might be just to get your system working on a limited number of different releases of the same operating system running on the same processor architecture. Or perhaps you've been asked to make your Linux application run on AIX - a job filled with snake Dpits to trap the unwary, as anyone who's crossed swords with AIX will know. And at the extreme, you may be faced with cross-compiling from a variety of host operating systems to target multiple very different operating systems, GUIs, processor architectures and toolchains.

Judging by some of the open source libraries I've been working with recently, this isn't an area that's commonly done well. This aim of this session is to share techniques and experiences of organising and building multi-platform and cross-platform code. I'll be looking at the common approaches to multi-platform and cross-platform coding and how some build tools I've experienced deal with it.

There will also be a bit of a rant about GNU Autotools and the perils of system introspection. But mainly I want this to be a highly interactive session. I want to hear from you, and your experiences meeting the same challenge - what are the tools and techniques you use, what are their good and bad points? Let's improve our portability together.

Sat 13 Mar 09:30-11:00

Title	Oral Session
Presenter	Dietmar Kühl

2021

Processing Decimal Values

DIETMAR KÜHL[1]

[1] *Bloomberg LP, London, United Kingdom*

For typical interactions we are used to processing values represented in decimal. Computers are much more versatile processing using binary representations. As a result it is quite common that decimal values are processed using a binary presentation. Sadly, doing so does cause problems when fractional values are involved. This presentation explains the representation of floating points in a computer and analyses typical problems encountered when using binary floating points to represent decimal values. It then describes alternative representation, in particular decimal floating point and decimal fixed point, and why these solve the relevant problems. A decimal fixed point representation is the preferred approach when the number of fractional digits used in an application is known. However, that is often not the case. Thus, the design of a library implementation for decimal floating point is discussed.

Sat 13 Mar 09:30-11:00

Title	Oral Session
Presenter	James Pascoe

2021

C++20 + Lua = Flexibility

JAMES PASCOE[1]

[1] *Blu Wireless, Bristol, United Kingdom*

This talk describes an approach for combining C++20 with Lua. A key benefit of this combination is its flexibility i.e. performance critical features can be implemented in C++, whereas, behavioural aspects can be expressed in Lua. As Lua is interpreted, the program's behaviour can be changed on-the-fly or in the field without a compilation environment. Thus, hypotheses can be tested, workarounds can be explored and previously unknown requirements can be accommodated without requiring a full release cycle. This session presents the technical details for how to implement such an architecture. The aim of the session is to provide the audience with enough knowledge to be able to implement these ideas in their own projects. In particular, the session will show how to use SWIG to generate bindings and type mappings between C++20 and Lua, how to combine Lua coroutines (which are stateful) with stateless C++20 coroutines and how to integrate SWIG, Lua and C++20 into a CMake build flow. SWIG type mappings (with code available on GitHub) are provided for: std::span, std::any, std::optional and std::variant. This presentation is a significantly updated and expanded follow-on to a talk given at CppOnSea 2020. Audience members do not need to have seen the CppOnSea talk i.e. the session is completely standalone. However, those that have seen the CppOnSea presentation will benefit from the new content, in particular, the update to C++20, the new SWIG type mappings and a more comprehensive treatment of the interplay between C++ and Lua coroutines.
A further benefit is that these ideas have been tested in a large commercial deployment. As a running exemplar, the talk will describe how Blu Wireless (the author's employer) has migrated a mission-critical application from a monolithic C++98 code-base to the C++20/Lua architecture described here. In addition to the talk, the author will provide a C++20/Lua application called 'LuaChat' (available on GitHub) to consolidate the concepts and to provide a basis for audience experimentation.

Sat 13 Mar 11:30-13:00

Title	Oral Session
Presenter	Dmitry Kandalov

2021

Limited work-in-progress for developers

DMITRY KANDALOV[1]

[1] *Code Mine, London, United Kingdom*

The idea of limited work-in-progress (WIP) is coming from Lean methodologies. At its core it means that new tasks should only be started when the current piece of work is done and delivered. Finding the right work-in-progress limit can increase overall system (organisation) throughput. This idea can be applied on many levels including writing code. In this live coding session I will write FizzBuzzWoof code kata in Kotlin showcasing software development workflows which can be used for limiting work-in-progress. In particular: change size notifications, auto-revert, TDD, TCR (test && commit || revert). Learning outcomes: Details of workflows that help minimise work-in-progress while writing code. Prerequisites: Basic knowledge of a programming language similar to Java/Scala/Kotlin.

Sat 13 Mar 11:30-13:00

Title	Oral Session
Presenter	Lucian Radu Teodorescu

2021

Threads Considered Harmful

LUCIAN RADU R. TEODORESCU[1]

[1] *Garmin, Cluj-Napoca, Romania*

Multithreaded programming is everywhere nowadays. However, the way we construct multithreaded programs is still largely primitive; similar to using gotos in the era of structured programming. The current methods have usability, composability, correctness and also performance problems. This talk aims at providing a new toolset for the multithreaded programmer. Instead of writing our multithreaded programs in terms of explicit threads and synchronization primitives, one should be expressing them in terms of tasks and dependencies between tasks. The talk describes a method of modeling the programs and dependencies. It also shows how one can eliminate locks (or any synchronization primitives) from programs. Using this method, one can raise the abstraction of multithreaded programs, bring back composability, improve determinism, and

also improve performance. The talk will be a mix of theory (presented in an intuitive manner), comparisons with other models, practical examples, and, of course, performance considerations.

Sat 13 Mar 11:30-13:00

Title	Oral Session
Presenter	Erik Engheim

2021

A tour of Julia

ERIK ENGHEIM[1]

[1] *Sixty North, Oslo, Norway*

A high performance, just in time compiled dynamic language, which gives python like productivity with C/C++ style performance. We will get into the details of why this seemingly impossible task was made possible through clever language design. Then we will geek out by generating LLVM byte code and x86 assembly code for individual functions interactively at the command line, explore meta programming with Julia's LISP style macros and learn about Julia's powerful multiple dispatch mechanism and how it differs from function overloading. Julia is an upcoming language being embraced by the scientific and high performance computing community, running projects like Celeste on 650 000 cores processing 178 terabytes of astronomical data.

Sat 13 Mar 11:30-13:00

Title	Oral Session
Presenter	Roger Orr

2021

Let's look at lambdas

ROGER M. ORR[1]

[1] *OR/2 Limited, London, United Kingdom*

Lambda expressions first appeared in C++11 and have been extended slightly in each revision since then. I will be taking a look at lambdas and trying to answer questions such as these: - what are they? - when should I use them? - how do they work? - what are some of the pitfalls? I hope that anyone who knows C++ will be able to follow most of the content, and increase their understanding of this part of the C++ language!

Sat 13 Mar 11:30-13:00

Title	Oral Session
Presenter	Guy Davidson

2021

C++ and Linear Algebra

GUY DAVIDSON[1]

[1] *Creative Assembly, Hove, United Kingdom*

There is a glaring omission in the standard library: there is no built-in support for linear algebra. This has lead to languages like Python becoming ascendant in machine learning and artificial intelligence. This is a strange situation to find ourselves in. C++ is the default choice for high performance applications. For example, in the real-time rendering of 3D games, millions of linear algebra calculations are performed each second, and yet each game must create or choose a maths library for geometry and coordinate transformation. In this talk I will discuss the need for linear algebra, the typical requirements of a linear algebra library, and the proposal I am co-authoring to add linear algebra types and syntax to the standard library. I will present examples of simple applications in geometry and colour manipulation, and consider interaction with BLAS, the industry standard FORTRAN/C linear algebra library, with a view to implementing the new types and syntax. This is a new talk reflecting the current status of the proposal.

Sat 13 Mar 14:30-16:00

Title	Oral Session
Presenter	Kate Gregory

2021

Naming is Hard: Let's Do Better

KATE GREGORY[1]

[1] *Gregory Consulting, Ontario, Canada*

C++ developers are famously bad at naming: our idioms, guidelines, and lore are rich in examples of terrible names. For example, consider RAII, which stands for scope bound resource management, or west const which perhaps should be const west, or all the samples that feature an object called x which is an instance of a class called X, and so on. The good news is that naming well is a learned skill, and you can learn it, and start to name better right away. In this talk, I'll tell you why names matter, what benefits a good name can bring, and how to be better at naming. I'll discuss some categories of names and some common decisions within those categories. I'm not going to give you a set of rules to follow: this is about thinking and considering the meaning of the things you are naming. I will give you some questions to ask yourself and some structure that I use to help those who read what I write. I'll also address renaming things in existing (legacy) code, why and when to do it, and why getting it right the first time may not even be a realistic goal. You should be a lot more confident naming things after we spend this time together.

Sat 13 Mar 14:30-16:00

Title	Oral Session
Presenter	Nico Josuttis

2021

std::jthread - I told you concurrency is tricky

NICO JOSUTTIS[1]

[1] IT Communication, Braunschweig, Germany

With C++20 we will have a new basic thread class, std::jthread. It will fix a few flaws of std::thread, which was not designed as an easy to use RAII type and lacks the ability to support stopping a running thread. Sounds like we only have to implement a better destructor and add a parameter to signal cancellation. But especially with concurrency the devil is in the details. This is not just a talk about jthread. It is a talk about how implementing even pretty simply concurrency requirements can become a tricky task with many many traps.

Sat 13 Mar 14:30-16:00

Title	Oral Session
Presenter	Phil Nash

2021

Write Once, Run Mobile

PHIL NASH[1]

[1] JetBrains, Remote, United Kingdom

A cross-platform framework for mobile platforms has been the elusive holy grail ever since Android started to become competitive with iOS. Many attempts have been made: Xamarin, PhoneGap and, more recently, Flutter, to give some examples. While some of these have had some success, many shops that care about native experience are still keeping iOS and Android teams siloed and performing redundant work. This is still not a solved problem. So why would Kotlin/ Multi-platform be any different? We'll see how Kotlin/ MPP captures mindshare and familiarity, along with an architecture that fits the way teams prefer to work. We'll also dive a bit deeper into how Kotlin/ Native fits into this and how it works.

Sat 13 Mar 14:30-16:00

Title	Oral Session
Presenter	Pete Muldoon

2021

Retiring the Singleton Pattern, Concrete Suggestions for What to Use Instead

PETE MULDOON[1]

[1] Bloomberg LP, NY, The United States of America

"The worst part of this whole topic is that the people who hate singletons rarely give *concrete suggestions* for what to use instead." In this talk, we will explore just such an approach for replacing the *Singleton* pattern in large codebases. It is easy to slip into the pattern of creating singletons - particularly in *_large legacy code bases_* - where low level functions need to propagate side effects like database updates, IPC etc. + Passing parameters down long function call chains can be daunting in terms of scope of change required in a large codebase. + Additionally, users calling a long established API in _legacy code_ are frequently unwilling to change their calls to supplement the current data being passed in. After briefly reviewing a classic singleton approach to a typical problem – sending requests to a server – and it associated drawbacks. We will rework the example and replace the function's internal singleton calls with calls to an explicitly passed in wrapper class. The extra information required for legacy users is injected via a custom default instance of the wrapper so that the users of the original function require no coding changes. We will then show how the previously untestable function can now be subjected to unit testing via dependency injection. This idea is later expanded to cover * keeping ABI stable * dealing with non-copyable types * dealing with delayed construction * dealing with Singleton dependency groupings * Initialization order of interdependent singletons, replacing error prone explicit intialization ordering with hard to misuse automatic initialization driven by the language. * Showing how the replacement pattern can be gradually introduced to a large code base instead of 'all at once'. * statefull grouping of dependencies. * Configuring long-lived Singleton replacement Objects This alternative approach has been successfully employed in multiple areas in *Bloomberg* where developers believed there was no other feasible choice.

Sat 13 Mar 14:30-16:00

Title	Oral Session
Presenter	Mike Shah

2021

A Study of Plugin Architectures for supporting Extensible Software

MIKE SHAH[1]

[1] Northeastern University, Boston, The United States of America

It is becoming more and more common for users to also contribute to the development of the software that they use--especially in the domains of computer graphics and gaming. Terms like 'modding' software have been around since the early 90s when the popular game Doom allowed for users to create their own content and modify the behavior of the program. Behind these programs there thus must be a mechnanism for allowing users to 'hook' into the main program. In this talk, I will be showing several software developer kits including Autodesk Maya 3D (C++), Unity3D (C#), Unreal Engine (C++), and QT Modeler(C), and discuss present a case study of how they are designed. At the end of the design discussion I will present how to get started building your own plugin system, and what considerations must be taken in mind (e.g. does the application or plugin manage resources, what should be exposed in the API, how do you embed a scripting language, and how should you distribute your plugins). Attendees will leave the presentation with practical knowledge on how to deliver software that can be extended by their userbase.

Sat 13 Mar 16:30-18:00

Title	Invited Slot
Presenter	Sean Parent
Guest Presenter	Sean Parent

2021

Keynote: Better Code: Relationships

SEAN PARENT[1]

[1] Sean Parent, San Jose, CA, The United States of America

Computer scientists are bad at relationships. Nearly every program crash is rooted in a mismanaged relationship, yet we spend most of our time discussing types and functions and not the relationships connecting them together. This talk looks at common ways data and code are connected in an application, how those relationships are typically represented, and the problems caused by the use, and misuse of these paradigms. Then we'll look at ways to model these relationships in C++ and use them to build correct applications.

Sun 14 Mar 10:00-18:00

Title	Oral Session
Presenter	Nico Josuttis

2021

Modern C++ Template Programming

NICO JOSUTTIS[1]

[1] IT Communication, Braunschweig, Germany

Each and every C++ programmer uses templates. Containers such as vector<> or array<>, strings, algorithms such as sort(), iterators, and I/O streams are all implemented as generic code. Modern C++ adds type traits, smart pointers, template member functions such as emplace(), and generic lambdas as a tricky form of generic code. Nevertheless the knowledge and understanding of how to implement and use templates is very limited and each and every programmer is sooner or later getting lost. This workshop therefore discusses templates for a whole day to make clear what it means to use templates and how to use them in practice. The focus is on Modern C++ demonstrating the benefit of using the language features of C++11, C++14, and even C++17. As a result the general understanding of templates will be improved and generic code might become more helpful and less surprising. Outline: We go through the standard template topics (function templates, class templates, non-type templates, specialization and other tricky basics) and spice them with the modern language feature to do up-to-date template programming. Then special modern features come into play: move semantics, variadic tempates, fold expressions (C++17), class template argument deduction (C++17). Finally some special consequences: polymorphism with templates (including using std:variant<>), type utilities, SFINAE, and a bit on metaprogramming.

Sun 14 Mar 10:00-18:00

Title	Oral Session
Presenter	Mateusz Pusz

2021

C++ Concepts: Constraining C++ Templates in C++20 and Before

MATEUSZ PUSZ[1]

[1] Epam Systems | Train IT, Gdansk, Poland

C++ Concepts is one of the most significant and long-awaited features of C++20. They improve template interfaces by explicitly stating the compile-time contract between the user and the architect of the code, which limits the number of compilation errors and make them much more user-friendly when they occur. The workshop will describe this C++20 feature, its similarities and differences from Concepts TS (provided with gcc-7), and will present ways to benefit from a significant part of the functionality in current production C++ projects with the usage of "legacy" C++11 features.

Sun 14 Mar 10:00-18:00

Title	Oral Session
Presenter	Peter Sommerlad

2021

Good Modern C++ Design and Practices

PETER SOMMERLAD[1]

[1] *Better Software: Consulting, Training, Reviews, Wollerau, Switzerland*

This workshop is trying to simplify your use of C++. We have many great rule sets to chose from, some partially outdated, like Scott Meyers 3rd edition, some futuristic, like the C++ core guidelines. While working on the AUTOSAR C++ and new MISRA C++ guidelines I found that many of the guidelines forbid things without giving actual guideline on how to do things and when to deviate. Also many talks on C++ explain the modern features and show how they work, but only few put things into context and show what to give up and how things combine sanely. I am guilty of that in the past as well, e.g., with my constexpr compile time computation talks at ACCU. This full day workshop is the result of thinking about that. It won't show C++20 feature by feature, but gives a coherent set of practices to improve your design and code using existing standard C++ features where they give you benefits. We will cover the following topics: * designing function interfaces in a way that they are easy to call correctly and hard to call incorrectly * how to report function contract violations (at least 5 different ones) and their individual benefits and liabilities, so you can make a conscious choice. * what parameter passing style and return value style works best under what conditions * how to create (parameter) type wrappers to avoid passing wrong arguments * class design for simple value wrappers to improve function interfaces * mix-in strategies for functionality and operators, so that creating value wrappers is simpler * provide an overview of class styles, e.g., value, manager, oo-bases and show how to select from the rules for special member functions * take a look at the lesser known C++11 feature of ref-qualified member functions and show why and when to use them for your member functions If you are brave enough, bring your own examples that we can look at and discuss where they are perfect and where they could be improved. Otherwise, we will take a look at potential bugs in the C++ standard library design.

Tuesday 2021-03-09

10:00	Modern C++ Idioms **Mateusz Pusz**	Good Modern C++ Design and Practices **Peter Sommerlad**	Building and Packaging Modern C++ **Adrian Ostrowski** **Piotr Gaczkowski**	ACCU 101: Early Career Day **Gail Ollis** **Kevlin Henney** **Giovanni Asproni** **Chris Oldwood** **Roger Orr, Jon Skeet** **Arjan van Leeuwen** **Jez Higgins**
12:00	Better Code **Sean Parent**			

Wednesday 2021-03-10

09:00	Keynote: Technical Agile Coaching with the Samman method **Emily Bache** Track A				
10:30	BREAK				
11:00	How C++20 changes the way we write code **Timur Doumler** Track A	Dynamic Polymorphism with Code Injection and Metaclasses **Sy Brand** Track B	This Videogame Developer Used the STL and You'll Never Guess What Happened Next **Mathieu Ropert** Track C	Drawing for IT Architects **Filip Van Laenen** Track D	What does the linker actually do for us? **CB Bailey** **Andy Balaam** Track E
13:00	BREAK				
14:00	Safer C++: MISRA-C++:202x rules and beyond **Peter Sommerlad** Track A	The C++ rvalue lifetime disaster **Arno Schödl** Track B	Typical Type Typos **Amir Kirsh** Track C	TBD Track D	How technical debt can kill your business. How F1 teams crack technical debt. **Luca Minudel** Track E
15:30	BREAK				
16:00	A Practical Introduction to C++20's Modules **Hendrik Niemeyer** Track A	Future of testing with C++20 **Kris Jusiak** Track B	Reflection: Compile-time Introspection of C++ **Andrew Sutton** Track C	You can't test this? Hammertime! **Frances Buontempo** **Steve Love** Track D	Refactoring Superpowers: make your IDE do your work, faster and more safely **Clare Macrae** Track E

Thursday 2021-03-11

09:00	Keynote: It Depends... **Kevlin Henney** **Track A**				
10:30	BREAK				
11:00	An Overview of Standard Ranges **Tristan Brindle** **Track A**	Generic Programming without (writing your own) Templates **Tina Ulbrich** **Track B**	Windows, macOS and the Web: Lessons from cross-platform development at think-cell **Sebastian Theophil** **Track C**	Contrasting test automation and BDD: an “interactions over tools” perspective **Seb Rose** **Track D**	Interesting Characters **Andy Balaam** **Track E**
12:30	BREAK				
12:45	Undo Software Lunch & Learn Webinar: Time Travel Debugging - it’s time to Debug Different **Chris Croft-White** **Track A**				
14:00	**14:00-14:20** C++ UNIverse **Victor Ciura**	**14:00-14:20** Building portable C++ packages: the Curse of Abundance **Piotr Gaczkowski**		What use is a confined user shell? **Alan Griffiths** **Track D**	PowerShell for the Curious **Chris Oldwood** **Track E**
		14:20-14:40 Building portable C++ packages: the bliss of unification **Adrian Ostrowski**	**14:20-14:40** Programming as a sport – what do you mean? **Ahto Truu**		
	14:45-15:05 The Point Challenge - returning different types for the same operation **Amir Kirsh**	**14:45-15:05** Testing your tests with code coverage **Richard Wallman**	**14:45-15:05** Services evolution: required is forever **Natalia Pryntsova**		
	15:10-15:30 Example mapping: a structured, collaborative discovery technique **Seb Rose** **Track A**	**15:10-15:30** Tools that spark joy: lessons learned from the Rust ecosystem that can be adopted elsewhere **Lotte Steenbrink** **Track B**	**15:10-15:30** Handling large volumes of immutable structured data with ClickHouse **Jim Hague** **Track C**		

16:00 The Business	Value of a Good API **Bob Steagall** Track A	Cross-Platform Pitfalls and How to Avoid Them **Erika Sweet** Track B	Frictionless Allocators **Alisdair Meredith** Track C	Modern Linux C++ debugging tools - under the covers **Greg Law** **Dewang Li** Track D	Thinking in Immediate: ImGUI **Zhihao Yuan** Track E

Friday 2021-03-12

09:00 Keynote: Who are they, and what do they want?
Patricia Aas Track A

10:30 BREAK

11:00	AddressSanitizer on Windows **Victor Ciura** Track A	Playing with Security **Charles Weir** Track B	Contrasting Hideous Mathematics for the software engineer **Patrick Martin** Track C	API Vulnerabilties and What to Do About Them **Eoin Woods** Track D	From Iterators To Ranges — The Upcoming Evolution Of the Standard Library **Arno Schoedl** Track E

12:30 BREAK

14:00	JavaScript the Grumpy Parts **Rob Richardson** Track A	Rethinking the Way We Do Templates in C++ **Mateusz Pusz** Track B	Modern C and what we can learn from it **Luca Sas** Track C	Hi, I'm Dom, and I Have Depression! **Dom Davis** Track D	How to build digital signatures from hash functions **Ahto Truu** Track E

15:30 BREAK

16:00	C++ Concepts vs Rust Traits vs Haskell Typeclasses vs Swift Protocols **Conor Hoekstra** Track A	Lakos'20: The "Dam" Book is Done! **John Lakos** Track B	C++11/14 at scale - what have we learned? **Vittorio Romeo** Track C	Redesigning Legacy Systems - Strategies that work / Lessons learned **Pete Muldoon** Track D	API Vulnerabilties and What to Do About Them **Eoin Woods** Track E

20:00 Echoborg Entertainment **I am Echoborg**

Saturday 2021-03-13

09:30	Concurrency in C++20 and beyond **Anthony Williams** **Track A**	Processing Decimal Values **Dietmar Kühl** **Track B**	C++20 + Lua = Flexibility **James Pascoe** **Track C**	C++20 Templates - The next level: Concepts and more **Andreas Fertig** **Track D**	Building and organising a multi-platform development code base **Jim Hague** **Track E**
11:00	BREAK				
11:30	Threads Considered Harmful **Lucian Radu Teodorescu** **Track A**	Let's look at lambdas **Roger Orr** **Track B**	C++ and Linear Algebra **Guy Davidson** **Track C**	Limited work-in-progress for developers **Dmitry Kandalov** **Track D**	A tour of Julia **Erik Engheim** **Track E**
13:00	BREAK				
14:30	std::jthread - I told you concurrency is tricky **Nico Josuttis** **Track A**	Retiring the Singleton Pattern, Concrete Suggestions for What to Use Instead **Pete Muldoon** **Track B**	A Study of Plugin Architectures for supporting Extensible Software **Mike Shah** **Track C**	Naming is Hard: Let's Do Better **Kate Gregory** **Track D**	Write Once, Run Mobile **Phil Nash** **Track E**
16:00	BREAK				
16:30	Keynote: Better Code: Relationships **Sean Parent** **Track A**				

Sunday 2021-03-14

10:00	Modern C++ Template Programming **Nico Josuttis**	C++ Concepts: Constraining C++ Templates in C++20 and Before **Mateusz Pusz**	Good Modern C++ Design and Practices **Peter Sommerlad**

udb
by undo

Time's Up for Bugs

Time Travel Debugging takes the guesswork out of debugging. Capture and replay code execution history to get instant visibility into what your program just did and why.

- ✓ **Fix bugs faster**
- ✓ **Uncover root cause with 100% certainty**
- ✓ **Understand codebases you didn't write**

undo.io/udb

30-DAY FREE TRIAL

UDB: DEBUG DIFFERENT

2021 CONFERENCE SPEAKERS

Bache, Emily

Emily Bache is a Technical Agile Coach with ProAgile. She helps teams to improve their coding and testing skills, including Test-Driven Development. Emily lives in Gothenburg, Sweden, but is originally from the UK. She is the author of "The Coding Dojo Handbook" and often speaks at international conferences. twitter: @emilybache blog: http://coding-is-like-cooking.info/

Fullday Workshop: Getting High Regression Test Coverage Quickly using Approval Testing

Bailey, CB

CB is a software developer at Bloomberg. CB works in Bloomberg Application Services where they help application developers easily write and maintain software than integrates and communicates in

robust and efficient ways. CB's previous career in software has included roles in such diverse areas as web technology, business intelligence, data warehousing, defence and radar. CB understands the importance of optimal software practices and so has a keen interest in source control systems and best practices surrounding their use. CB is a Git user, advocate and contributor and relishes the opportunity to slice through knotty problems with their git-fu and to teach others how to do the same.

Session: What does the linker actually do for us?

Balaam, Andy

Andy Balaam loves code, and loves talking about code. His blog, articles and open source projects can be found at http://artificialworlds.net and his videos are at http://youtube.com/ajbalaam

Session: What does the linker actually do for us?

Balaam, Andy

Andy is happy as long as he has a programming language and a problem. He finds over time he has more and more of each. You can find his open source projects at artificialworlds.net or contact him on mail@artificialworlds.net

Session: Interesting Characters

Brand, Sy

Sy is Microsoft's C++ Developer Advocate. Their background is in compilers and debuggers for embedded accelerators, but they're also interested in generic library design, metaprogramming, functional-style C++, undefined behaviour, and making our communities more welcoming and inclusive.

Session: Dynamic Polymorphism with Code Injection and Metaclasses

Brindle, Tristan

Tristan is a freelance developer, C++ trainer and BSI committee member based in London. He's the author of NanoRange, a C++17-compatible Ranges implementation, and lead tutor for C++ London Uni, a not-for-profit organisation offering free weekly C++ classes for students in London and around the world.

Session: An Overview of Standard Ranges

Buontempo, Frances

Frances Buontempo is currently editor of the ACCU's Overload magazine and is a programmer by profession. She has a BA in maths and philosophy, an M.Sc. in Pure Mathematics, and a PhD in data mining to predict how toxic organic chemicals might be. Between then and now, she has worked in various companies in Leeds and London with a finance focus or as a consultant. She has talked and written about various ways to program your way out of a paper bag, providing a gentle introduction to some machine learning approaches, while trying to keep up to date with new techniques. She wrote these up in a book [https://pragprog.com/book/fbmach/genetic-algorithms-and-machinelearning-for-programmers]

Session: You can't test this? Hammertime!

Ciura, Victor

Victor Ciura is a Principal Engineer at CAPHYON, Technical Lead on the Advanced Installer team and a Microsoft MVP (Developer Technologies). He's a regular guest at Computer Science Department of his Alma Mater, University of Craiova, where he gives student lectures & workshops on using C++ STL Algorithms. Since 2005, he has been designing and implementing several core components and libraries of Advanced Installer. Currently, he spends most of his time working with his team on improving and extending the repackaging and virtualization technologies in Advanced Installer IDE, helping clients migrate their traditional desktop apps to the modern Windows application format: MSIX. One of his "hobbies" is tidying-up and modernizing the aging codebase of Advanced Installer and has been known to build tools that help this process: Clang Power Tools More details: @ciura_victor & https://ciura.ro

Session: C++ UNIverse

Session: AddressSanitizer on Windows

Davidson, Guy

Guy Davidson has been developing in C++ for over 30 years and writing games for nearly 40. He is the Principal Coding Manager at the UK's largest games studio, Creative Assembly, makers of Total War, Alien:Isolation, Halo Wars 2 and others, where he helps good programmers become better programmers. He has been there for 20 years and shows no signs of slowing down. He is the coauthor of the linear algebra library proposal, as well as the audio library proposal and the 2D graphics library proposal, among others. He hopes to bring HMI to the standard and works with SG13 and SG14 to achieve this.

Session: C++ and Linear Algebra

Davis, Dom

Dom Davis is a veteran of The City and a casualty of The Financial Crisis. Not content with bringing the world to its knees he then went off to help break the internet before winding up in Norfolk where he messes about doing development and devops. Dom has been writing code since his childhood sometime in the last millennium – he hopes some day to become good at it. Dom is an enthusiastic and impassioned speaker [read: he gabbles] who uses a blend of irreverent sarcasm and flippant humour to bring complex subjects to a broad audience. Whether or not they understand him is up for debate, but he likes to believe they do.

Session: Hi, I'm Dom, and I Have Depression!

Doumler, Timur

Timur Doumler is a C++ developer specialising in audio and music technology, active member of the ISO C++ committee, and part of the includecpp.org team. He is passionate about building communities, clean code, good tools, and the evolution of C++.

Session: How C++20 changes the way we write code

Engheim, Erik

Erik Engheim has been programming for the last two decades in a variety of programming languages primarily C/C++ but also Java, C#, Objective-C and Swift. He is the author of the "Getting Started with Julia" video course on the new programming language Julia used in high performance and scientific computing. He also has a passion for crypto currencies, UX design, space exploration, green technologies, robotics and micro controllers. Erik has worked in a variety of industries: Oil & Gas, Fintech, Video conferencing and IT consulting.

Session: A tour of Julia

Ertsås, Martin

Martin Ertsås is a software developer working for Cisco Systems in Norway on their Telepresence Hardware Endpoints. His main interests are C++, Linux, Security, Embedded Systems, and Developer Happiness. Martin enjoys digging through new code to unravel how it works, or spending time improving a tool or process to increase the happiness and productivity of those around him.

Fullday Workshop: Creating a sandbox for you Linux Application

Fertig, Andreas

Andreas Fertig is the CEO of Unique Code GmbH, which offers training and consulting for C++ specialized in embedded systems. He worked for Philips Medizin Systeme GmbH for ten years as a C++ software developer and architect focusing on embedded systems. Andreas is involved in the C++ standardization committee. He is a regular speaker at conferences internationally. Textbooks and articles by Andreas are available in German and English. Andreas has a passion for teaching people how C++ works, which is why he created C++ Insights (cppinsights.io).

Session: C++20 Templates - The next level: Concepts and more

Gaczkowski, Piotr

Music and automation enthusiast. Focused on efficiency and effectiveness. Experienced in management, programming, and DevOps. Enjoys building simple solutions to human problems. Writes occasionally at https://doomhammer.info. Speaks of himself in the third person when required. Never without headphones around. Rarely without sunglasses.

Session: Building portable C++ packages: the Curse of Abundance

Fullday Workshop: Building and Packaging Modern C++

Gregory, Kate

Kate Gregory has been using C++ for over thirty years. She writes, teaches, mentors, codes, and leads projects, primarily in C++. Kate is a Visual C++ MVP, has written over a dozen books, and speaks at conferences and user groups around the world. Kate develops courses on C++, Visual Studio, and Windows programming for Pluralsight, is active on over a dozen StackExchange sites, blogs infrequently, and is happy to be part of C++ Twitter and the #include Discord server.

Session: Naming is Hard: Let's Do Better

Griffiths, Alan

Alan is an experienced and effective proponent of the craft of software development. Interested in development processes, tools, design and coding techniques. He has a BSC in Mathematics and has published articles in ACCU's Overload and C Vu, C/C++ Users Journal, Java Report, and EXE. Contributor to "97 Things Every Programmer Should Know". His expertise covers a range of programming languages, tools and platforms. Although he has used many other programming languages over the years, he keeps returning to C++. Alan is leading a team of open source developers on "Mir" (a new Linux display server - https://mir-server.io) and working with "snaps" (a way of packaging applications for confinement - https://snapcraft.io). He has been Chair of the ACCU and a member of the BSI C++ Panel.

Session: What use is a confined user shell?

Hague, Jim

Jim learnt C from first edition K&R, bought the first edition of The C++ Programming Language when it first appeared, and hasn't stopped using either since. This has taken him over time through all sorts of environments, from JVM internals to air traffic control. He is currently nesting in the DNS world, and running a Code Club in his spare time.

Session: Building and organising a multi-platform development code base

Session: Handling large volumes of immutable structured data with ClickHouse

Henney, Kevlin

Kevlin is an independent consultant, trainer, reviewer and writer. His development interests are in programming, people and practice. He has been a columnist for various magazines and web sites, a contributor to open source software and a member of more committees than is probably healthy (it has been said that "a committee is a cul-de-sac down which ideas are lured and then quietly strangled"). He is co-author of two volumes in the Pattern-Oriented Software Architecture series and editor of 97 Things Every Programmer Should Know and the forthcoming 97 Things Every Java Programmer Should Know. blog: https://medium.com/@kevlinhenney

Session: Keynote: It Depends...

Hoekstra, Conor

Conor Hoekstra is a Senior Library Software Engineer at NVIDIA working on the RAPIDS team. He is extremely passionate about programming languages, algorithms and beautiful code. He is the founder and organizer of the Programming Languages Virtual Meetup and he has a YouTube channel.

Session: C++ Concepts vs Rust Traits vs Haskell Typeclasses vs Swift Protocols

Josuttis, Nico

Nicolai Josuttis is well known in the programming community because he not only speaks and writes with authority (being the (co-)author of the world-wide best sellers The C++ Standard Library (www.cppstdlib.com), C++ Templates (www.tmplbook.com), C++17 - The Complete Guide (www.cppstd17.com), and SOA in Practice), but is also an innovative presenter, having talked at various conferences and events. He is an independent system architect, technical manager, author, and consultant. He designs mid-sized and large software systems for the telecommunications, traffic, finance, and manufacturing industries.

Session: std::jthread - I told you concurrency is tricky

Fullday Workshop: Modern C++ Template Programming

Jusiak, Kris

Kris is a Senior Software Engineer passionate about programming and who has worked in different industries over the years including telecommunications, games and most recently finance for Quantlab Financial, LLC. He has an interest in modern C++ development with a focus on performance and quality. He is an open-source enthusiast with multiple open-source libraries where he uses template meta-programming techniques to support the C++ rule - "Don't pay for what you don't use" whilst trying to be as declarative as possible with a help of domain-specific languages. Kris is also a keen advocate of extreme programming techniques, Test/Behavior Driven Development and truly believes that 'the only way to go fast is to go well!'.

Session: Future of testing with C++20

Kandalov, Dmitry

Dmitry has been programming since DOS times. He spent the last 15 years or so in Java lands most recently working with server-side Kotlin.

Session: Limited work-in-progress for developers

Kirsh, Amir

C++ lecturer at the Academic College of Tel-Aviv-Yaffo and at Tel-Aviv University. Previously the Chief Programmer at Comverse. Expert in C++, software design and development in general.

Session: The Point Challenge - returning different types for the same operation

Session: Typical Type Typos

Kühl, Dietmar

Dietmar Kühl is a senior software developer at Bloomberg L.P. working on the data distribution environment used both internally and by enterprise installations at clients. Before joining Bloomberg he has done mainly consulting for software projects in the finance area. He is a regular attendee of the ANSI/ISO C++ standards committee, presents at conferences, and he used to be a moderator of the newsgroup comp.lang.c++.moderated. He frequently answers questions on Stackoverflow.

Session: Processing Decimal Values

Lakos, John

John Lakos, author of Large-Scale C++ Software Design, serves at Bloomberg LP in New York City as a senior architect and mentor for C++ Software Development world-wide. He is also an active voting member of the C++ Standards Committee's Evolution Working Group. Previously, Dr. Lakos directed the design and development of infrastructure libraries for proprietary analytic financial applications at Bear Stearns. For 12 years prior, Dr. Lakos developed large frameworks and advanced ICCAD applications at Mentor Graphics, for which he holds multiple software patents. His academic credentials include a Ph.D. in Computer Science ('97) and an Sc.D. in Electrical Engineering ('89) from Columbia University. Dr. Lakos received his undergraduate degrees from MIT in Mathematics ('82) and Computer Science ('81). His new book, the first volume of which is entitled Large-Scale C++ — Volume I: Process and Architecture (2020), is now available from Pearson Education.

Session: Lakos'20: The "Dam" Book is Done!

Law, Greg

Greg is the co-founder and CTO of Undo. He has over 20 years' experience in the software industry and has held development and management roles at companies including the pioneering British computer firm Acorn, as well as fast-growing start ups, NexWave and Solarflare. It was at Acorn that Greg met Julian and on evenings and weekends, they invented the core technology that would eventually become UndoDB. From the beginnings in his garden shed, Greg led Undo to a 50-person company based in Cambridge and San Francisco, until in 2018 he became full-time CTO. Greg holds a PhD from City University, London and was nominated for the 2001 British Computer Society Distinguished Dissertation Award. He lives in Cambridge, UK with his wife and two children. In his spare time, Greg catches up on email.

Session: Modern Linux C++ debugging tools - under the covers

Li, Dewang

DeWang is a Solutions Architect at Synopsys's Software Integrity Group. He is passionate about making source code more robust and secure through static analysis and other technologies. DeWang actively works with the world's top programmers in Silicon Valley, including at places like Amazon AWS, Tesla, and nVidia. He believes source code is a core asset, and teaming up with programmers to ensure quality, security, and maintainability is a noble goal.

Session: Modern Linux C++ debugging tools - under the covers

Love, Steve

Steve Love has never written a compiler, but once wrote a tiny operating system of which he was very proud at the time. He still considers himself to be a programmer, despite spending much of his time talking about testing and deployment instead of actually writing code.

Session: You can't test this? Hammertime!

Macrae, Clare

Clare is an independent consultant, helping teams work sustainably and efficiently to test and refactor legacy and hard-to-test code. She has worked in software development for over 30 years, and in C++ and Qt for 20 years, and is now branching out to other languages. Since 2017, Clare has used her spare time to work remotely with Llewellyn Falco on https://github.com/approvals/ApprovalTests.cpp[ApprovalTests.cpp], to radically simplify testing of legacy code. She has enjoyed this so much that in 2019 she set up Clare Macrae Consulting Ltd, to focus even more on helping others work with legacy code. Before this, Clare was a Principal Scientific Software Engineer at Cambridge Crystallographic Data Centre. She is the original author of their popular C++ and Qt-based 3D crystal structure visualisation program https://www.ccdc.cam.ac.uk/mercury/[Mercury].

Session: Refactoring Superpowers: make your IDE do your work, faster and more safely

Martin, Patrick

Patrick's github repo was classified using a machine learning gadget as belonging to a 'noble corporate toiler'. He can't top that.

Session: Hideous Mathematics for the software engineer

Meredith, Alisdair

Alisdair Meredith is a software developer at BloombergLP in New York, and a previous chair of the C++ Standard Committee Library Working Group He has been an active member of the C++ committee for almost two decades, and by a lucky co-incidence his first meeting was the kick-off meeting for the project that would become C++11, and also fixed the contents of the original library TR. He is currently working on the BDE project, BloombergLP's open source libraries that offer a foundation for C++ development, including a standard library implementation supporting the polymorphic allocator model that was ultimately adopted by C++17.

Session: Frictionless Allocators

Minudel, Luca

Luca Minudel is a Lean-Agile Coach & Trainer, and a Transformation lead, with 20+ years of experience in professional software delivery and digital product development, most of them with Lean and Agile. He is passionate about agility, lean, complexity science, and co-creation. He contributed to the adoption of lean and agile practices by Ferrari's F1 racing team. For ThoughtWorks he delivered training, coaching, assessments and organisational transformations in top-tier organisations in Europe and the United States. He worked as Head of Agility, Agile Transformation Lead, Lean/Agile Practice Lead, and as Lean/Agile Coach in companies such as HSBC, Lloyds, LexisNexis. Luca is the founder and CEO at SmHarter.com, a company that helps organisations turn their way of working into their competitive advantage.

Session: How technical debt can kill your business. How F1 teams crack technical debt.

Muldoon, Pete

Pete Muldoon has been using C++ since 1991. Pete has worked in Ireland, England and the USA and is currently employed by Bloomberg. A consultant for over 20 years prior to joining Bloomberg, Peter has worked on a broad range of projects and code bases in a large number of companies both tech and finance. Such broad exposure has, over time, shown what works and what doesn't for large scale engineering projects. He's a proponent of elegant solutions and expressive code.

Session: Retiring the Singleton Pattern, Concrete Suggestions for What to Use Instead

Session: Redesigning Legacy Systems - Strategies that work / Lessons learned

Nash, Phil

Phil is the author of the C++ test framework, Catch2, and composable command line parser, Clara and has recently taken an interest in Kotlin/Native, too. As Developer Advocate at JetBrains he's involved with CLion, AppCode and ReSharper C++. More generally he's an advocate for good testing practices, TDD and using the type system and functional techniques to reduce complexity and increase correctness. He's previously worked in Finance and Mobile as well as an independent consultant and coach specialising in TDD on iOS.

Session: Write Once, Run Mobile

Niemeyer, Hendrik

Hendrik is a System Architect and works on the software architecture for machine learning and big data applications. His favorite programming languages, in which he also has the most experience, are C++ and Rust. He described himself as a "learning enthusiast" who always gets absorbed in trying out new things.

Session: A Practical Introduction to C++20's Modules

Oldwood, Chris

Chris is a freelance programmer who started out as a bedroom coder in the 80's writing assembler on 8-bit micros. These days it's enterprise grade technology in plush corporate offices. He also commentates on the Godmanchester duck race and can be easily distracted via gort@cix.co.uk or @chrisoldwood.

Session: PowerShell for the Curious

Orr, Roger

Roger has many years of experience in IT, using a variety of languages and platforms, working for a number of different companies over the years, mostly in the financial sector. His recent work has mostly been in C++, on both Windows and Linux. Roger is one of the organisers of this conference and also runs the Code Critique column in ACCU's "CVu" magazine. He is chair of the UK C++ panel, has represented the UK at C++ ISO standards meetings since 2010, and is a member of the 'Direction Group', a five person group that recommends priorities for the ISO C++ standardisation committee.

Session: Let's look at lambdas

Ostrowski, Adrian

Modern C++ enthusiast, interested in the newest language standards and development of highquality code. Previously promoting music bands as a member of the board for the Kompresor foundation, as well as C++ at EPAM as a member of the board for its C++ Community. Previously working on a commodity exchange's trading system, currently working on the architecture of Intel and Habana's integration with machine learning frameworks. Fullday Workshop: Building and Packaging Modern C++

Session: Building portable C++ packages: the bliss of unification

Parent, Sean

Sean Parent is a senior principal scientist and software architect for Adobe's mobile digital imaging group and Photoshop. Sean has been at Adobe since 1993 when he joined as a senior engineer working on Photoshop and later managed Adobe's Software Technology Lab. In 2009 Sean spent a year at Google working on Chrome OS before returning to Adobe. From 1988 through 1993 Sean worked at Apple, where he was part of the system software team that developed the technologies allowing Apple's successful transition to PowerPC.

Session: Keynote: Better Code: Relationships

Fullday Workshop: Better Code

Pascoe, James

James Pascoe is a Principal Software Team Leader at Blu Wireless in Bristol. Blu Wireless builds mmWave mobile wireless IP links for high-speed transport and fixed wireless applications. At Blu Wireless, James is responsible for the software that exists above the MAC, primarily, the Blu Wireless Linux driver and the mobility software agent that makes decisions about which access point to connect to and when. Prior to Blu Wireless, James was a Senior Engineer at Intel where he worked on the Android graphics stack. Prior to Intel, James held various hardware and software positions at STMicroelectronics and prior to ST, James was a Post Doctoral Research Fellow at the Department of Math and Computer Science at Emory University in Atlanta, GA. James hold a first class degree and a PhD from the University of Reading (both in Computer Science) and an MBA (with distinction) from Warwick University.

Session: C++20 + Lua = Flexibility

Polce, Paolo

Paolo Polce has 20+ years of professional software development experience across industries including: Formula 1 Motorsport, Pharmaceutical, Media Publishing and more. His focus is system simplification, continuous refactoring and delivery. Paolo works at "Simul Works" (http://www.simulworks.com)

Session: How technical debt can kill your business. How F1 teams crack technical debt.

Pryntsova, Natalia

Natalia Pryntsova is a team leader in Bloomberg L.P. with particular interest in distributed systems design. Before joining Bloomberg she did software consulting work on variety of projects in finance and has seen both good and bad architectural practices in action. Day to day she mostly uses Python and C++ but still secretly admires C#.

Session: Services evolution: required is forever

Pusz, Mateusz

A software architect, chief engineer, and security champion with more than 15 years of experience in designing, writing and maintaining C++ code for fun and living. C++ consultant, trainer, conference speaker, and evangelist focused on Modern C++. His main areas of interest and expertise are code performance, low latency, stability, and security. Mateusz worked at Intel for 13 years, and now he is the head of the C++ Competency Center at EPAM Systems. He is also a founder of Train IT that provides dedicated C++ trainings and consultant services to corporations. Mateusz is a contributor and an active voting member of the ISO C++ Committee (WG21) where, together with the best C++ experts in the world, he shapes the future of the C++ language. He is also a co-chair of WG21 Study Group 14 (SG14) responsible for driving performance and low latency subjects in the Committee. In 2013 Mateusz won "Bench Games 2013" – worldwide competition in the C++ language knowledge.

Fullday Workshop: C++ Concepts: Constraining C++ Templates in C++20 and Before

Session: Rethinking the Way We Do Templates in C++

Fullday Workshop: Modern C++ Idioms

Richardson, Rob

Rob Richardson is a software craftsman building web properties in ASP.NET and Node, React and Vue. He's a Microsoft MVP, published author, frequent speaker at conferences, user groups, and community events, and a diligent teacher and student of high quality software development. You can find this and other talks on https://robrich.org/presentations and follow him on twitter at @rob_rich.

Session: JavaScript the Grumpy Parts

Romeo, Vittorio

Vittorio Romeo (B.Sc. Computer Science) has been a Software Engineer at Bloomberg for more than 3 years, working on mission-critical company C++ infrastructure and providing Modern C++ training to hundreds of fellow employees. He began programming around the age of 8 and quickly became a C++ enthusiast. Vittorio created several open-source C++ libraries and games, published many video courses and tutorials, and actively participates in the ISO C++ standardization process. He is also an active member of the C++ community and has an ardent desire to share his knowledge and learn from others: he presented more than 20 times at international C++ conferences (including CppCon, C++Now, ++it, ACCU, C++ On Sea, C++ Russia, and Meeting C++), covering topics of various nature. Vittorio maintains a website with advanced C++ articles and a YouTube channel featuring wellreceived modern C++11/14 tutorials. Lastly, he's active on StackOverflow, taking great care in answering interesting C++ question (60k reputation). When he's not writing code, Vittorio enjoys weightlifting and fitness-related activities, competitive/challenging computer gaming and sci-fi movies/TV-series.

Session: C++11/14 at scale - what have we learned?

Ropert, Mathieu

French C++ expert working on (somewhat) historical video games. Decided to upgrade his compiler once and has been blogging about build systems ever since. Past speaker at CppCon, Meeting C++ and ACCU. Used to run the Paris C++ User Group. Currently lives in Sweden.

Session: This Videogame Developer Used the STL and You'll Never Guess What Happened Next

Rose, Seb

Consultant, coach, trainer, analyst, and developer for over 30 years. Seb has been involved in the full development lifecycle with experience that ranges from architecture to support, from BASIC to Ruby. He's a BDD advocate with SmartBear, helping people integrate all three practices of BDD into their development process and ensuring that appropriate tool support is available. Regular speaker at conferences and occasional contributor to software journals. Co-author of the BDD Books series "Discovery" and "Formulation" (Leanpub), lead author of "The Cucumber for Java Book" (Pragmatic Programmers), and contributing author to "97 Things Every Programmer Should Know" (O'Reilly). He blogs at cucumber.io/blog and tweets as @sebrose.

Session: Contrasting test automation and BDD: an "interactions over tools" perspective

Session: Example mapping: a structured, collaborative discovery technique

Sas, Luca

Luca Sas is a Core Systems Engineer at Creative Assembly who has been coding for almost a decade and is passionate about system architecture and low level programming. Some of his previous work includes mobile apps with some of the biggest NGOs in Romania and video game development. In Romania he was a national champion at programming contests and olympiads where he is now a judge. He enjoys attending conferences and talking to developers about their experience and learning about ways to improve software design as well as mentoring new programmers and giving talks. Previously he gave talks at programming events in Romania and the University of Leeds, and ACCU 2019.

Session: Modern C and what we can learn from it

Schoedl, Arno

Arno Schödl, Ph.D. Founder & CTO Arno is responsible for the design, architecture and development of all our software products. He oversees think-cell's R&D team, Quality Assurance and Customer Care. Before founding think-cell, Arno worked at Microsoft Research and McKinsey. Arno studied computer science and management and holds a Ph.D. from the Georgia Institute of Technology with a specialization in Computer Graphics.

Session: The C++ rvalue lifetime disaster

Session: From Iterators To Ranges — The Upcoming Evolution Of the Standard Library

Shah, Mike

Mike currently is an Assistant Teaching Professor at Northeastern University. Along with his research on performance, he also consults as a Senior 3D Graphics Engineer on a variety of multimedia projects. Mike discovered computer science at the age of 13 when googling "how do I make games". From that google search, Mike has worked as a freelance game developer, worked in industry for Intel, Sony PlayStation, Oblong Industries, and researched at The Ohio Supercomputer Center to name a few. Mike cares about building tools to help programmers monitor and improve the performance of realtime applications – especially games.

Session: A Study of Plugin Architectures for supporting Extensible Software

Sommerlad, Peter

Peter Sommerlad is a consultant and trainer for Safe Modern C++ and Agile Software Engineering. Peter was professor at and director of IFS Institute for Software at FHO/HSR Rapperswil, Switzerland until February 2020. Peter is co-author of POSA Vol.1 and Security Patterns. He inspired the C++ IDE Cevelop with a unique C++ feedback, refactoring, and code modernization experience. Peter is a member of MISRA-C++, Hillside, ACM, IEEE Computer Society, ACCU, ISO WG23 and the ISO WG21 C++ committee.

Fullday Workshop: Good Modern C++ Design and Practices

Session: Safer C++: MISRA-C++:202x rules and beyond

Steagall, Bob

Bob Steagall has been working in C++ since discovering the second edition of "The C++ Programming Language" in a college bookstore in 1992. The majority of his career has been spent in medical imaging, where he led teams building applications for functional MRI and CT-based cardiac visualization. After a brief detour through the worlds of DNS and analytics, he's now working in the area of distributed stream processing. He is a voting member of the C++ Standardization Committee, and has a blog where he occasionally writes about C++ and related topics. Bob holds BS and MS degrees in Physics, is an avid cyclist when weather permits, and lives in fear of his wife's cats.

Session: The Business Value of a Good API

Steenbrink, Lotte

Lotte Steenbrink is an Embedded Software Engineer working at Ferrous Systems who has transitioned from C to C++ to now Rust. In the past, she has worked on Industrial IoT applications, networking for constrained devices, and protocol standardization (IETF). While picking up Rust, she's gotten involved in knurling-rs, an open source project on a mission to improve the embedded Rust development experience through better tooling. As a programmer she appreciates seemingly simple solutions to difficult problems, and as an impatient human she is fond of tools that support her in building what she needs and don't get in the way otherwise.

Session: Tools that spark joy: lessons learned from the Rust ecosystem that can be adopted elsewhere

Sutton, Andrew

Andrew Sutton is an owner of Lock3 Software, LLC where he designs languages, language features, and works on various compilers. Most of his work is focused on the C++ programming language.

Some of Andrew's current projects include the GCC implementation of C++ concepts, designing and implementing static reflection and metaprogramming for C++ using Clang, and the design and implementation of new programming languages. Andrew was formerly a university professor and taught undergraduate courses on programming with C++, programming languages, and compiler design.

Session: Reflection: Compile-time Introspection of C++

Sweet, Erika

Erika works on the Visual C++ Team at Microsoft. She likes math and mystery novels. She is currently working on developer tools to support C++ cross-platform development.

Session: Cross-Platform Pitfalls and How to Avoid Them

Teodorescu, Lucian Radu

Lucian Radu Teodorescu has a PhD in programming languages and is a Software Architect at Garmin. As hobbies, he is working on his own programming language and he is improving his Chuck Norris debugging skills: staring at the code until all the bugs flee in horror.

Session: Threads Considered Harmful

Theophil , Sebastian

Sebastian Theophil studied Computer Science in Berlin and Toulouse, and holds a PhD in Computer Science from Humboldt University of Berlin. He has been working at think-cell Software since its founding in 2002, and has recently been working on porting think-cell to the Mac.

Session: From Iterators To Ranges — The Upcoming Evolution Of the Standard Library

Session: Windows, macOS and the Web: Lessons from cross-platform development at think-cell

Session: The C++ rvalue lifetime disaster

Truu, Ahto

During his two and a half decades in the ICT industry, Ahto has worked in hardware installations and user support, as a software developer and architect, and as a systems analyst. Currently he is busy helping Guardtime's customers preserve the integrity of their important data. Outside his day job he coaches Estonia's team to the high school students' programming competitions. He has also been writing programming columns for the popular science magazines A&A and Horisont.

Session: How to build digital signatures from hash functions

Session: Programming as a sport -- what do you mean?

Ulbrich, Tina

Tina works at Rosen, a service provider in the oil and gas industry. She writes and maintains numerical and data processing algorithms for pipeline inspection data. She highly values simple, modern and clean code, using the latest language features. She promotes refactoring, high test coverage and collaboration between developers. She also teaches modern C++ in internal tech talks. Tina holds a university degree in Bio-Mathematics from the University of Applied Science in Zittau/ Goerlitz. She is an active member of the #include Discord community.

Session: Generic Programming without (writing your own) Templates

Van Laenen, Filip

Filip van Laenen started his career at Computas in the previous millennium, first as a Java developer, later as a business analyst, IT architect, and various other project roles. He's guilty of lots of bad drawings, but tries to improve himself, and would like to make the world a better place by helping others to improve too.

Session: Drawing for IT Architects

Wallman, Richard

Richard has been a developer for several decades, working on systems in VHDL and Verilog all the way up to JavaScript. During that time he's battled against errors and edge-cases in many programming languages, and as such has developed a strong appreciation for solid development practices. In 2006 he took on his biggest public project yet - designing and building the new platform for The Freecyle Network. Creating a system capable of handling millions of users and delivering a quarter of a billion emails every month, but within a non-profit's budget, required a ruthless approach towards security, efficiency and stability. Richard is currently working for a web-streaming company that handles the live webcasting of council meetings for UK councils such as Birmingham and Westminster. Working as the leader of PHP, JavaScript and C++ teams, he gets to deal with a variety of day-to-day coding issues, but it also allows him to cross-pollinate best practices between teams.

Session: Testing your tests with code coverage

Weir, Charles

Charles Weir is passionate about improving the security skills of teams of professional software developers. A researcher at Lancaster University, he designs interventions to help developers produce more secure software. His 'Developer Security Essentials' workshops have been used with development teams in a wide range of different organisations, and rigorously proven to have positive effects in every case. Previously he set up the mobile application development company, Penrillian, and ran it successfully for 15 years, employing up to thirty people and with a total turnover well over £30M. Charles also co-authored the book 'Small Memory Software', helped introduce objectoriented and agile methods to the UK, and was technical lead for the world's first smartphone.

Session: Playing with Security

Williams, Anthony

Anthony Williams is the author of C++ Concurrency in Action, and a UK-based developer, consultant and trainer with over 20 years of experience in C++. He has been an active member of the BSI C++ Standards Panel since 2001, and is author or coauthor of many of the C++ Standards Committee papers that led up to the inclusion of the thread library in the C++11 Standard. He continues to work on new facilities to enhance the C++ concurrency toolkit, both with standards proposals, and implementations of those facilities. Anthony lives in the far west of Cornwall, England, where he currently spends most of his time developing software for robots.

Session: Concurrency in C++20 and beyond

Woods, Eoin

Eoin Woods is CTO at Endava, where he guides technical strategy, oversees capability development and directs investment in emerging technologies. Eoin is a widely published author in both the research and industrial communities and a regular conference speaker, with expertise in software architecture, software security and distributed systems.

Session: API Vulnerabilties and What to Do About Them

Yuan, Zhihao

Zhihao Yuan is an HPC Engineer at SimpleRose Inc. He participated in standardizing designated initializers and improved narrowing conversions in C++20. In the past few months, he enjoyed writing Python programs in Visual Studio and avoided configuring another Vim emulation layer. He loves the Utawarerumono game series so much and is playing them on PS4 again.

Session: Thinking in Immediate: ImGUI