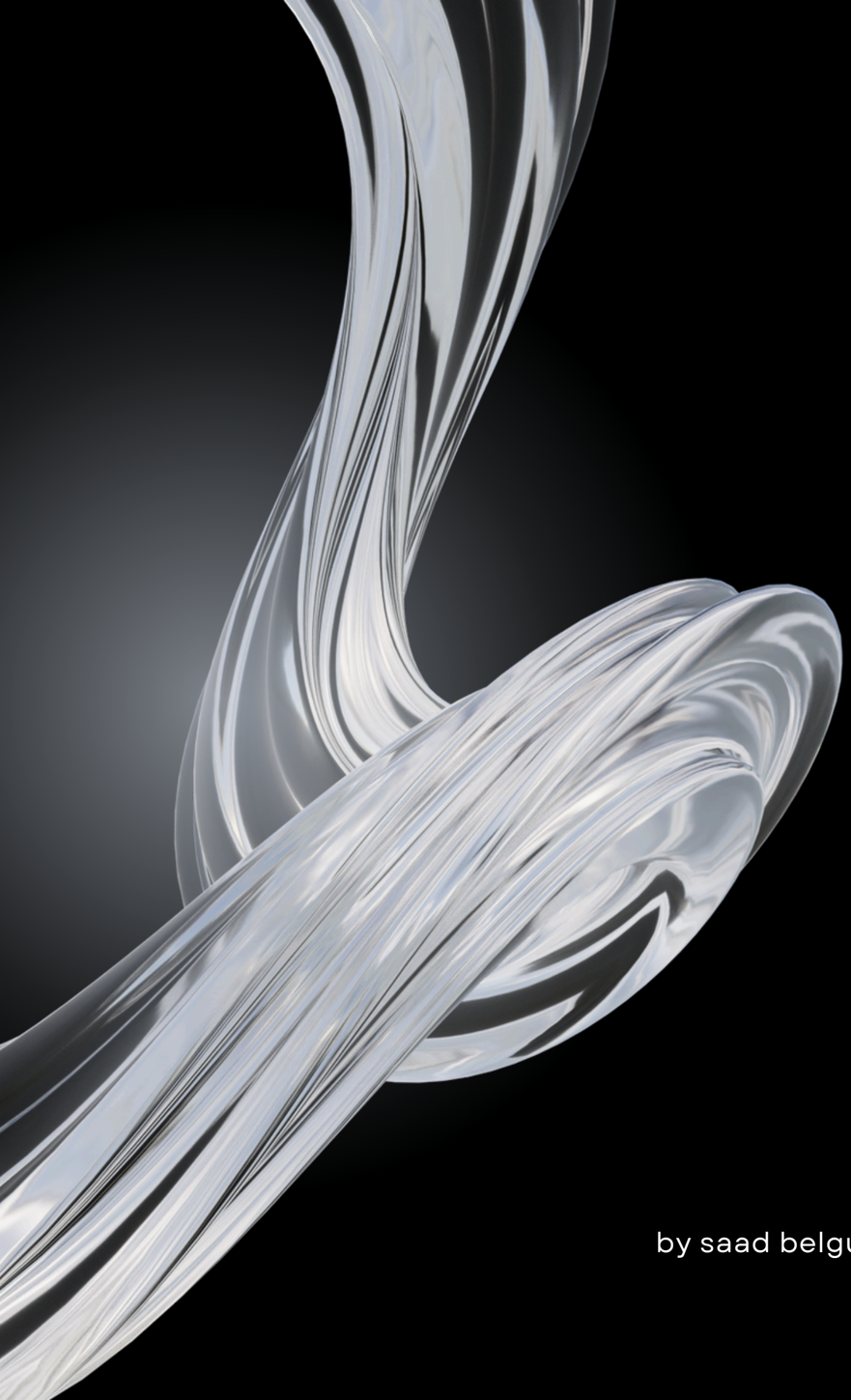
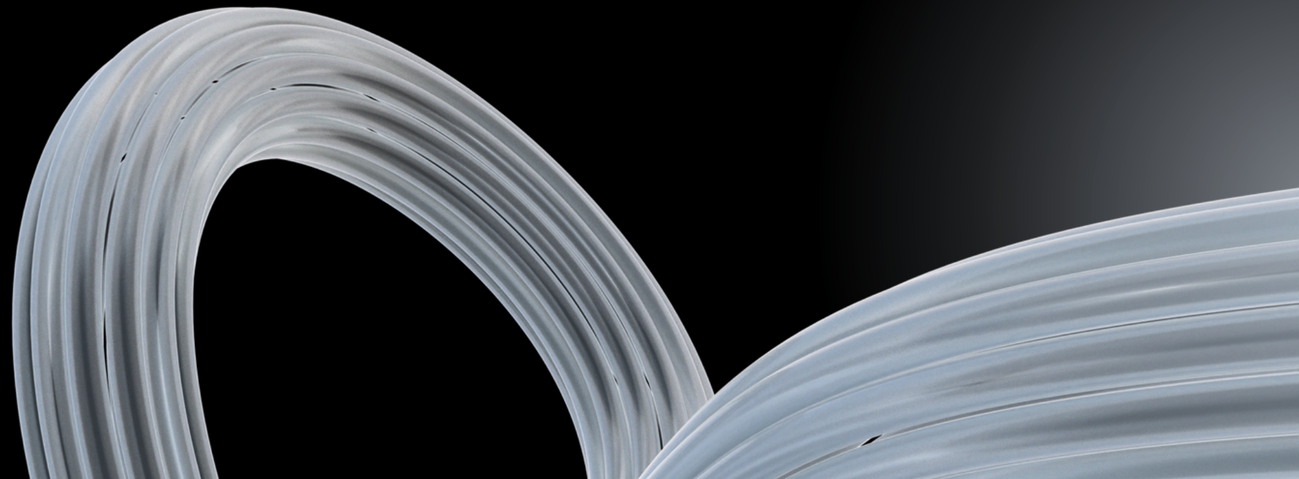
2024 U.S. FEDERAL TAX CALCULATOR

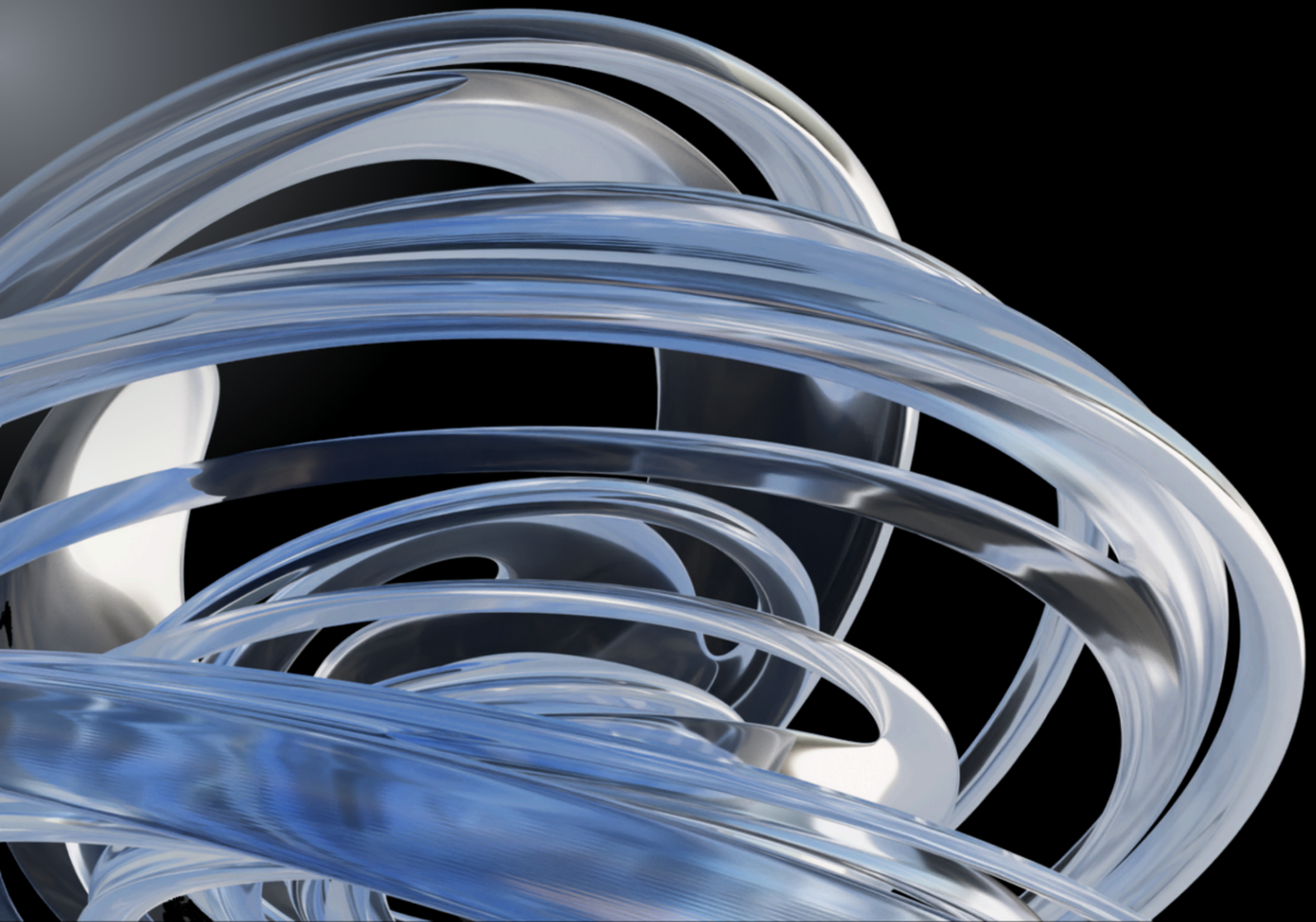
A MODULAR C++ CONSOLE APPLICATION





by saad belgued

CONTEXT



Application Purpose:

- Automates U.S. federal income tax calculations
- Designed for tax year 2024
- Computes AGI, MAGI, deductions, and tax liability
- Console-based for accessibility

Why It Matters:

- Tax calculation is complex and error-prone
- Provides accurate, instant results
- Educational tool for learning tax concepts

PROBLEM ADDRESSED

Key Challenges:

- Multiple filing statuses (5 different categories)
- Progressive tax brackets (7 levels per status)
- Complex deductions (standard, age-based, blindness)
- Adjustments to income (above-the-line deductions)

Solution:

- Automated calculation system
- Reduces manual errors
- Saves time for taxpayers
- Educational resource

TARGET USERS

Primary Users:

- U.S. taxpayers preparing 2024 returns
- Accounting students
- Financial literacy learners

Secondary Users:

- Software developers studying C++
- Educators in finance/computer science
-

Accessibility:

- No GUI required
- Runs on any system with C++ compiler
- Simple command-line interface

ARCHITECTURE OVERVIEW

```
tax-calculator/
├── header/                  # All header files
│   ├── taxpayer.hpp         # TaxPayer class + FilingStatus enum
│   ├── income.hpp           # Income data structure
│   ├── adjustments.hpp      # Adjustments data structure
│   ├── AGI.hpp              # AGI calculator interface
│   ├── magi.hpp             # MAGI calculator interface
│   ├── standard_deduction.hpp# Standard deduction calculator
│   ├── tax_bracket.hpp      # Tax bracket calculator
│   └── tax_constants_2024.hpp# IRS values for 2024
├── functions/               # Implementation files
│   ├── income.cpp           # Income implementation
│   ├── adjustments.cpp      # Adjustments implementation
│   ├── agi.cpp              # AGI calculation
│   ├── magi.cpp             # MAGI calculation
│   ├── standard_deduction.cpp# Standard deduction
│   └── tax_bracket.cpp      # Tax bracket calculation
└── main.cpp
```

DETAILED ARCHITECTURE

Data Layer:

- TaxPayer: Filing status, age, blindness

- Income: 7 income types (wages, interest, etc.)
- Adjustments: 8 deduction types (IRA, student loans, etc.)

Calculation Layer:

- AGI: Adjusted Gross Income
- MAGI: Modified Adjusted Gross Income
- StandardDeduction: With age/blindness bonuses
- TaxBracket: Progressive tax calculation

Constants:

- Tax2024: IRS values for 2024 tax year

Architecture Benefits & Design Choices

Why This Architecture Works:

1. Maintainability:
   - Each module can be updated independently
   - Adding new tax year = update constants only
   - Easy debugging - isolate issues to specific classes
2. Extensibility:
   - Add new income sources → update Income class only
   - Add new deductions → update Adjustments class only
   - Support new filing status → update enum and calculators
3. Testability:
   - Unit test each calculator independently
   - Mock data classes for testing
   - Verify IRS constants separately
4. Scalability:
   - Can add GUI layer without changing calculations
   - Can add database without changing business logic
   - Can add new tax credits as separate modules

Design Decisions Explained:

- No inheritance used → Simple composition
- Static methods → No state needed in calculators
- Header/CPP separation → Clear interface/implementation
- Enum classes → Type safety for filing status

LIBRARIES USED

Standard C++ Libraries:

- <iostream>: User input and output
- <vector>: Store tax brackets and rates
- <string>: Status conversions and display

No External Dependencies:

- Pure C++17 standard
- Cross-platform compatibility
- Easy compilation

PROGRAMMING ELEMENTS

Object-Oriented Design:

- Classes for each tax concept
- Encapsulation of related data
- Static methods for calculations

Q3

Key Features:

- Enum classes for type safety
- Const-correctness
- Header/implementation separation
- Modular testing capability

DETAILED ARCHITECTURE COMPONENTS

```cpp
#pragma once
#include <string>

class Income {
public:
    // Data members
    double wage= 0.0;
    double interest_income= 0.0;
    double dividend_income= 0.0;
    double unemployment_compensation= 0.0
    double retirement_distribution= 0.0;
    double other_income= 0.0;
    double business_income= 0.0;

    double getTotal() const;
};
```

```cpp
#include "../header/AGI.hpp"

// calculation of Adjusted Gross Income.
double AGI::calculate(const Income& income, const Adjustments& adjustments) {
    return income.getTotal() - adjustments.getTotal(); // AGI = Total Income - Adjustments
}

enum class FilingStatus {
    Single,
    MarriedFilingJointly,
    HeadOfHousehold,
    MarriedFilingSeparately,
    QualifyingWidower
};

inline std::string filingStatusToString(FilingStatus status) {
    switch (status) {
        case FilingStatus::Single:                return "Single";
        case FilingStatus::MarriedFilingJointly:  return "Married Filing Jointly";
        case FilingStatus::HeadOfHousehold:       return "Head of Household";
        case FilingStatus::MarriedFilingSeparately:return "Married Filing Separately";
        case FilingStatus::QualifyingWidower:     return "Qualifying Widow(er)";
        default:                                  return "Unknown Status";
    }
}
```

CHALLENGES & SOLUTIONS

Challenge 1: no itemized deductions

- Solution: adding itemized deduction
- Benefit: new method for those who want itemized

Challenge 2:no tax credits

- Solution: create tax credits
- Benefit: for those who benifit from those credits

CHALLENGES & SOLUTIONS

Challenge 1: no itemized deductions

- Solution: adding itemized deduction
- Benefit: new method for those who want itemized

Challenge 2:no tax credits

- Solution: create tax credits
- Benefit: for those who benifit from those credits

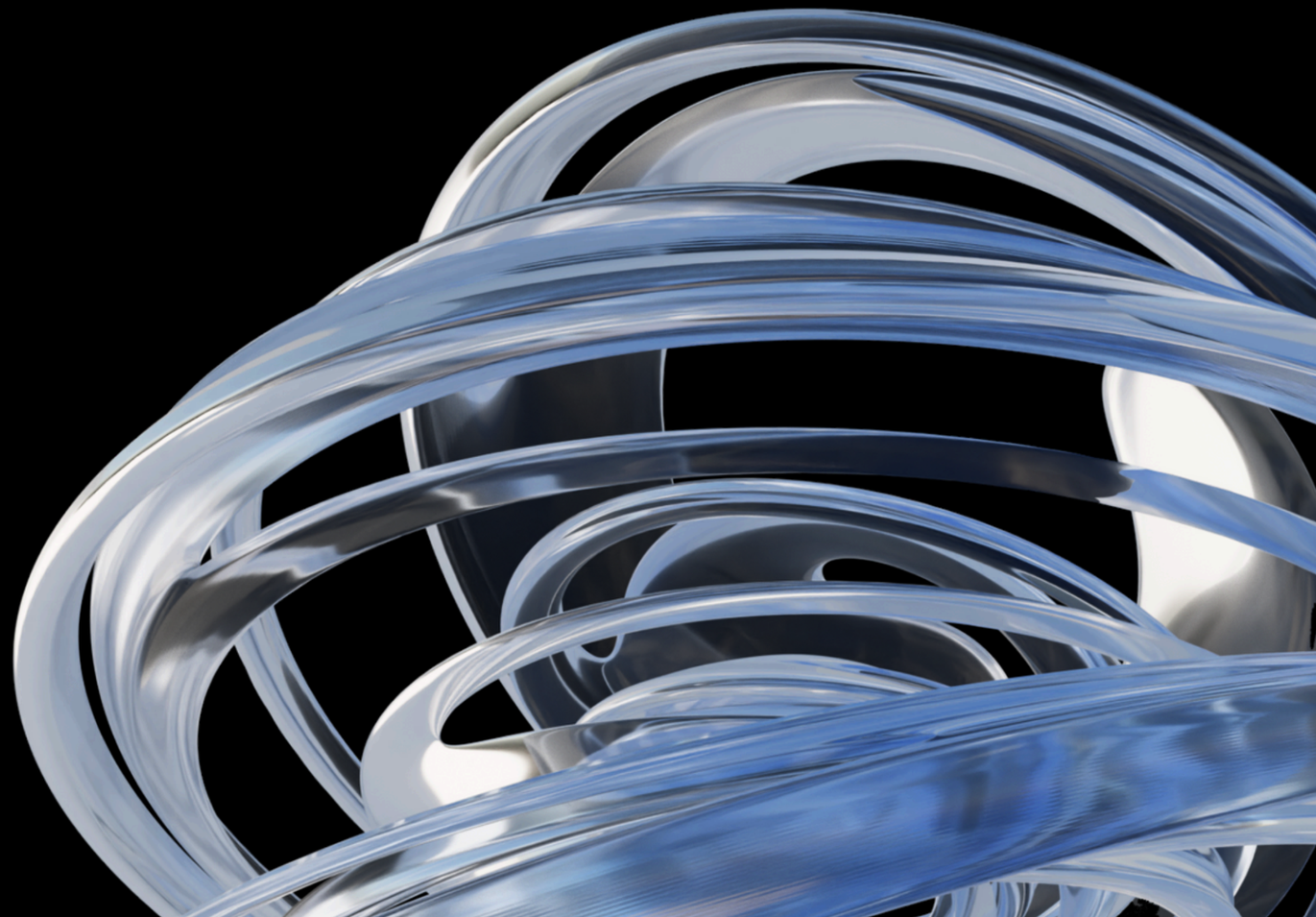
CURRENT LIMITATIONS

Scope Limitations:

- Only 2024 tax year (hardcoded)
- Standard deduction only (no itemized)
- No tax credits included
- No state tax calculations

Technical Limitations:

- Console interface only
- No input validation
- No data persistence
- No error recovery



FUTURE ENHANCEMENTS

Priority 1: GUI Implementation

- Qt or wxWidgets interface
- Real-time calculation updates
- Form-based input validation

Priority 2: Expanded Features

- Itemized deductions
- Tax credits (child, education, etc.)
- Multiple tax years (2023-2025)
- State tax calculations

Priority 3: Advanced Features

- PDF report generation
- Database storage
- Multi-user support
- Tax planning scenarios

THANK YOU