# Programiranje 2, letnji semestar 2023/4

# Uvodno predavanje

Stefan Nikolić

Prirodno-matematički fakultet, Novi Sad
19.02.2024.

# Šta je cilj ovog predmeta?

Da obezbedi dobre osnove programiranja u jeziku C++

# Zašto nam je bitno programiranje?

# Zašto nam je bitno programiranje?

yahoo!finance Search for news, symbols or companies Sign in Mail

Finance Cryptocurrencies News Property Money Yahoo Finance Invest Asia Markets Industries Watchlists My Portfolio My Screeners

**A programmer wrote scripts to secretly automate a lot of his job — including to automatically email his wife and make himself a latte**

Julie Bort
23 November 2015

And the best one? He wrote a script that waits exactly 17 seconds, then hacks into the coffee machine and orders it to start brewing a latte. The script tells the machine to wait another 24 seconds before pouring the latte into a cup, the exact time it takes it takes to walk from the guy's desk to the coffee machine.

And his coworkers didn't even know the coffee machine was on the network and was hackable.

Jer nam život čini lakšim

# Zašto nam je bitno programiranje?

Dobro, to je ipak sporedno...

# Zašto nam je bitno programiranje?

Ključno je to što nam može pomoći u rešavanju raznih matematičkih problema

# Zašto nam je bitno programiranje?

BULLETIN OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 82, Number 5, September 1976

## RESEARCH ANNOUNCEMENTS

### EVERY PLANAR MAP IS FOUR COLORABLE[1]

BY K. APPEL AND W. HAKEN

Communicated by Robert Fossum, July 26, 1976

The following theorem is proved.

THEOREM. *Every planar map can be colored with at most four colors.*

### EVERY PLANAR MAP IS FOUR COLORABLE PART I: DISCHARGING[1]

BY

K. APPEL AND W. HAKEN

The complexity of Osgood's work convinced Haken that, even with a further improved discharging procedure, the general case could not be effectively attacked without the aid of electronic computers for the tedious task of discussing all possible configurations which can be obtained by merging pairs, triplets,

# Zašto nam je bitno programiranje?

Massachusetts Institute of Technology

Education Research Innovation Admissions + Aid Campus Life News

**MIT News**

ON CAMPUS AND AROUND THE WORLD

SUBSCR

## The answer to life, the universe, and everything

Mathematics researcher Drew Sutherland helps solve decades-old sum-of-three-cubes puzzle, with help from "The Hitchhiker's Guide to the Galaxy."

Sandi Miller | Department of Mathematics
September 10, 2019


Booker devised an ingenious algorithm and spent weeks on his university's supercomputer when he recently came up with a solution for 33. But when he turned to solve for 42, Booker found that the computing needed was an order of magnitude higher and might be beyond his supercomputer's capability. Booker says he received many offers of help to find the answer, but instead he turned to his friend Andrew "Drew" Sutherland, a principal research scientist in the Department of Mathematics. "He's a world's expert at this sort of thing," Booker says.

Sutherland, whose specialty includes massively parallel computations, broke the record in 2017 for the largest Compute Engine cluster, with 580,000 cores on Preemptible Virtual Machines, the largest known high-performance computing cluster to run in the public cloud.

$$x^3 + y^3 + z^3 = 42$$

$$x^3 + y^3 + z^3 = 33$$

A. R. Booker *Res. Number Theory* (2019) 5:26
https://doi.org/10.1007/s40993-019-0162-1

Research in Number Theory

RESEARCH

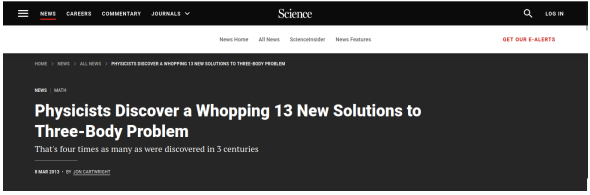
## Cracking the problem with 33

Andrew R. Booker

### 3 Implementation

We implemented the above algorithm in C, with a few inline assembly routines for Montgomery arithmetic [14] written by Buhrow [4], and Walisch's `primesieve` library [19] for enumerating prime numbers.

# Zašto nam je bitno programiranje?




The discovery of 13 new families, made by physicists Milovan Šuvakov and Veljko Dmitrašinović at the Institute of Physics Belgrade, brings the new total to 16. "The results are beautiful, and beautifully presented," says Richard Montgomery, a mathematician at the University of California, Santa Cruz, who was not involved with the discovery.

It's the sort of abstract puzzle that keeps a scientist awake at night: Can you predict how three objects will orbit each other in a repeating pattern? In the 300 years since this "three-body problem" was first recognized, just three families of solutions have been found. Now, two physicists have discovered 13 new families. It's quite a feat in mathematical physics, and it could conceivably help astrophysicists understand new planetary systems.

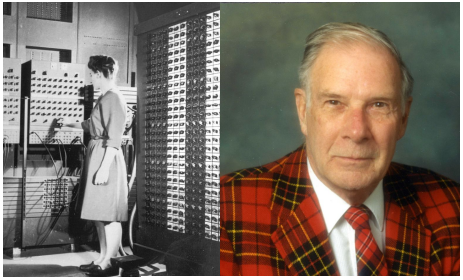
The trove of new solutions has researchers jazzed. "I love these things," says Robert Vanderbei a mathematician at Princeton University who was not involved in the work. He says he, in fact, spent all night thinking about the work.

UNIVERSITY OF BELGRADE | INSTITUTE OF PHYSICS | BELGRADE
NATIONAL INSTITUTE OF THE REPUBLIC OF SERBIA

# Zašto nam je bitno programiranje?

Ja sam matematičar/ka. Zar ne mogu ja da osmislim algoritam a da ga neko drugi implementira?

# Zašto nam je bitno programiranje?



**Richard Hamming**

**``You and Your Research''**

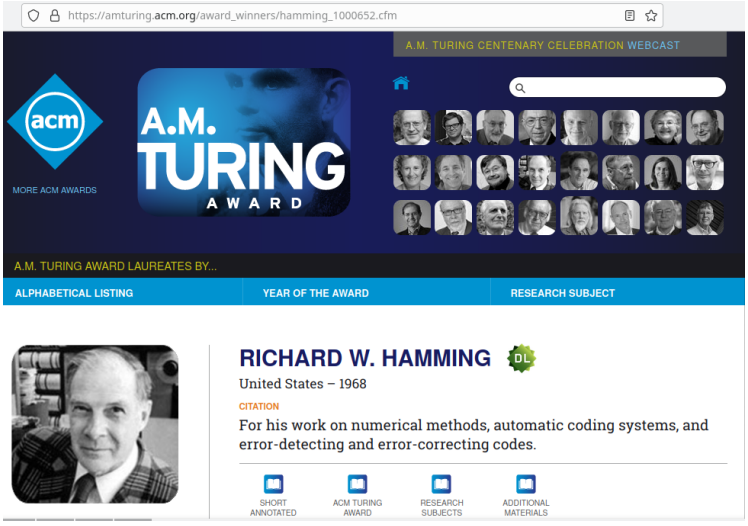
Transcription of the
Bell Communications Research Colloquium Seminar
7 March 1986

I give you a story from my own private life.
Early on it became evident to me that Bell Laboratories was not going to give me the conventional acre of programming people to program computing machines in absolute binary.

***

I finally said to myself, "Hamming, you think the machines can do practically everything. Why can't you make them write programs?" What appeared at first to me as a defect forced me into automatic programming very early.
What appears to be a fault, often, by a change of viewpoint, turns out to be one of the greatest assets you can have.
But you are not likely to think that when you first look the thing and say, "Gee, I'm never going to get enough programmers, so how can I ever do any great programming?"

# Zašto nam je bitno programiranje?



# Zašto nam je bitno programiranje?

Mogućnost da sami isprobate svoje ideje će vam dati veliku slobodu. Takođe, vi najbolje poznajete svoj algoritam, pa uz malo programerske veštine, možete napisati i najbolju implementaciju.

# Od Hemingovih dana je programiranje postalo jednostavnije

Džon Mokli i Presper Ekert
(John Mauchly & Presper Eckert)

ENIAC (Univerzitet u Pensilvaniji, 1945)

Ketlin But (Kathleen Booth)

Prvi asemblij jezik i prvi assembler
(Univerzitet u Londonu, 1947)

Džon Bakus (John Backus)

Fortran

Prvi široko-rasprostranjeni kompajlirani jezik
(IBM, 1957)

Ken Tomson i Denis Riči
(Ken Thompson & Dennis Ritchie)

Programski jezik u kom je napisan
operativni sistem Unix (Bell Labs, 1972)

Bjarne Stroustrup

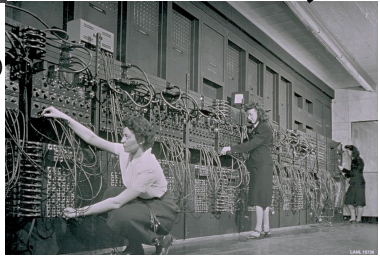
Objektno-orijentisano proširenje
programskog jezika C (Bell Labs, 1985)

Gvido van Rosum
(Guido van Rossum)

Python: trenutno najpopularniji
programski jezik na svetu (1991)

# Od Hemingovih dana je programiranje postalo jednostavnije

Džon Mokli i Presper Ekert
(John Mauchly & Presper Eckert)

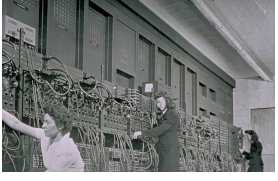


ENIAC (Univerzitet u Pensilvaniji, 1945)

Ketlin But (Kath

Prvi asembli jezik i
(Univerzitet u Lo

# Od Hemingovih dana je programiranje postalo jednostavnije



Ketlin But (Kathleen Booth)

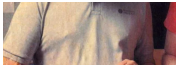
Prvi asembli jezik i prvi asembler
(Univerzitet u Londonu, 1947)

Fortran

Prvi široko-rasprostranjeni komp
(IBM, 1957)

# Od Hemingovih dana je programiranje postalo jednostavnije

Džon Bakus (John Backus)



Ken Tomson i De
(Ken Thompson & De

C

Fortran

AUTOMATIC CODING SYSTEM FOR THE IBM 704

Prvi široko-rasprostranjeni kompajlirani jezik
(IBM, 1957)

# Od Hemingovih dana je programiranje postalo jednostavnije

Ken Tomson i Denis Riči
(Ken Thompson & Dennis Ritchie)

John Backus)


Programski jezik u kom je napisan
operativni sistem Unix (Bell Labs, 1972)

# Od Hemingovih dana je programiranje postalo jednostavnije

Gvido van Rosu
(Guido van Rossu

Bjarne Stroustrup

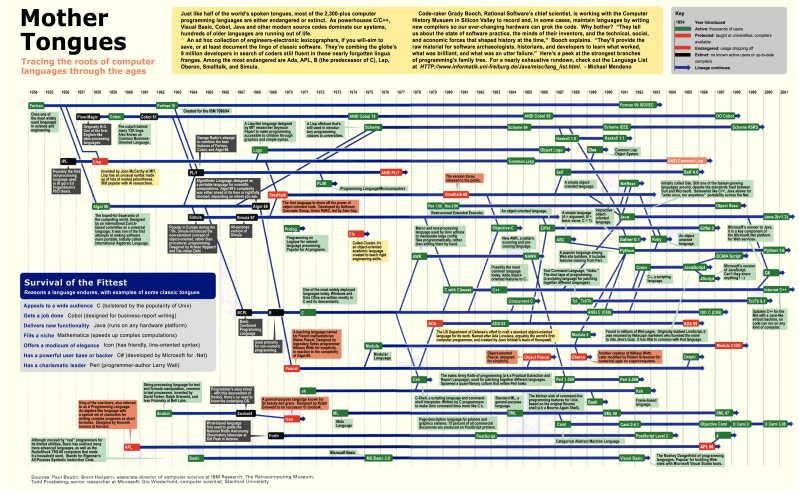
Objektno-orijentisano proširenje
programskog jezika C (Bell Labs, 1985)

# Od Hemingovih dana je programiranje postalo jednostavnije

Gvido van Rosum
(Guido van Rossum)

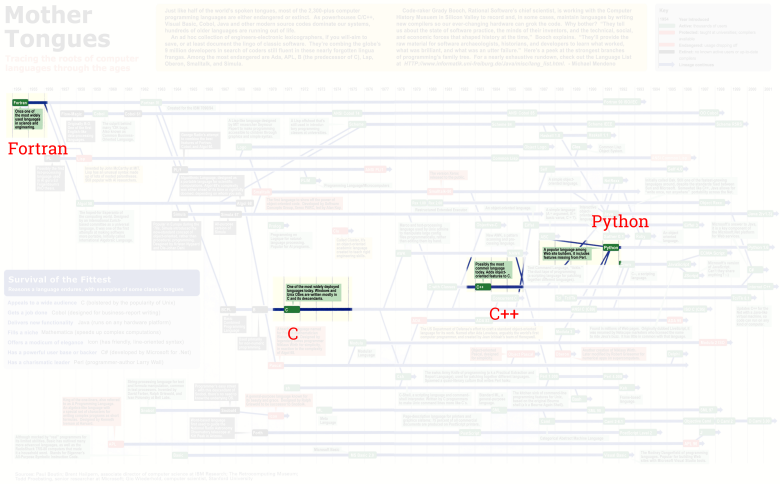
Python: trenutno najpopularniji programski jezik na svetu (1991)

# Ovo je samo delić...



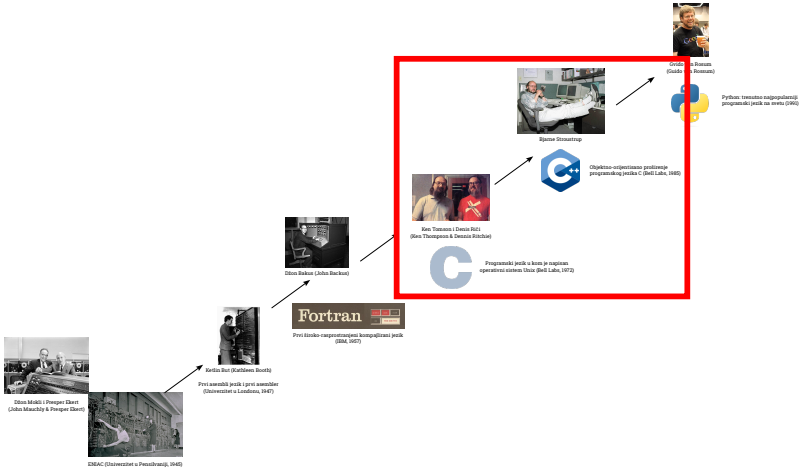

Izvor: https://ccrma.stanford.edu/courses/250a-fall-2005/docs/ComputerLanguagesChart.png
(graf predstavlja stanje do 2005)

# Ovo je samo delić...



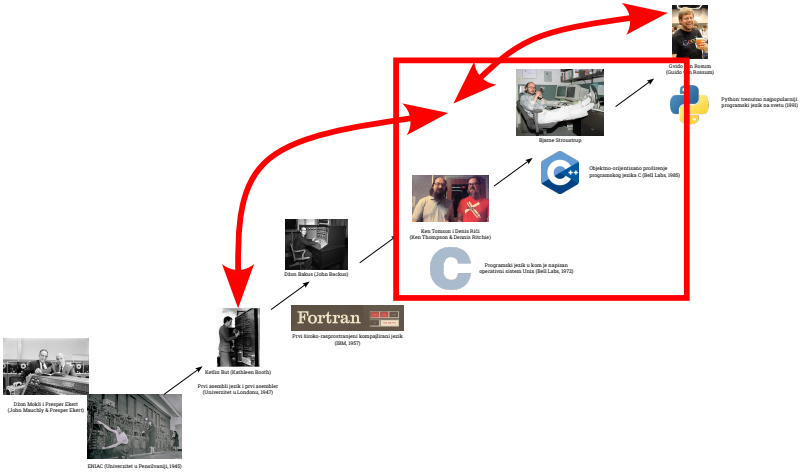

Izvor: https://ccrma.stanford.edu/courses/250a-fall-2005/docs/ComputerLanguagesChart.png
(graf predstavlja stanje do 2005)

# Na ovom predmetu ćemo učiti




C/C++

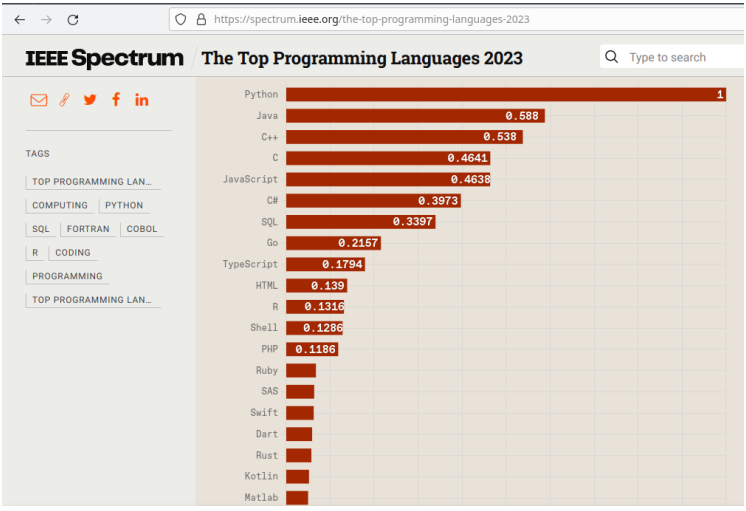
# Na ovom predmetu ćemo učiti




C/C++ i nešto malo o njihovoj interakciji sa asemblijem i Python-om

# Zašto baš C++?

# Zašto baš C++?




C++ je još uvek jedan od najrasprostranjenijih programskih jezika

# Zašto baš C++?

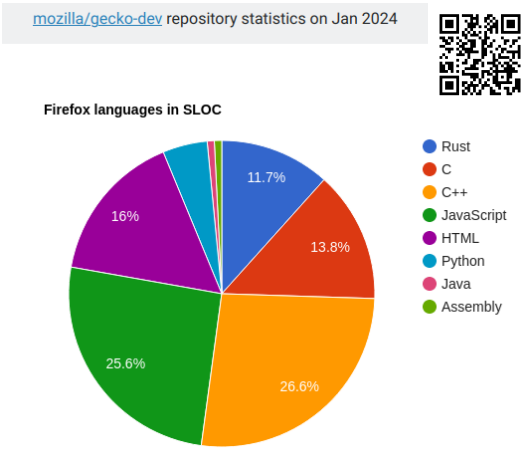
| Dialects |
| --- |
| Cyclone, Unified Parallel C, Split-C, Cilk, C* |
| **Influenced by** |
| B (BCPL, CPL), ALGOL 68,[4] PL/I, FORTRAN |
| **Influenced** |
| Numerous: AMPL, AWK, csh, C++, C--, C#, Objective-C, D, Go, Java, JavaScript, JS++, Julia, Limbo, LPC, Perl, PHP, Pike, Processing, Python, Rust, Seed7, V (Vlang), Vala, Verilog (HDL),[5] Nim, Zig |

Izvor: Vikipedija

| **Influenced by** |
| --- |
| Ada, ALGOL 68,[2] BCPL,[3] C, CLU,[2] F#,[4][note 1] ML, Mesa,[2] Modula-2,[2] Simula, Smalltalk[2] |
| **Influenced** |
| Ada 95, C#,[5] C99, Carbon, Chapel,[6] Clojure,[7] D, Java,[8] JS++,[9] Lua,[10] Nim,[11] Objective-C++, Perl, PHP, Python,[12] Rust,[13] Seed7 |

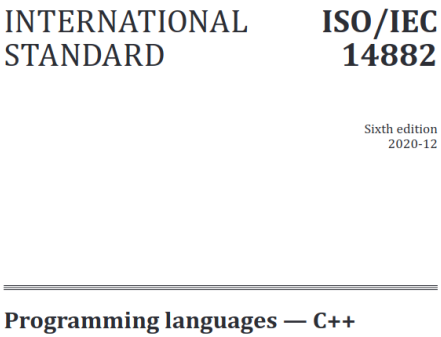
C i C++ leže u osnovi velikog broja drugih značajnih jezika

# Zašto baš C++?




U C/C++-u je napisana ogromna količina koda koji je u svakodnevnoj upotrebi

# Zašto baš C++?


INTERNATIONAL STANDARD

ISO/IEC 14882

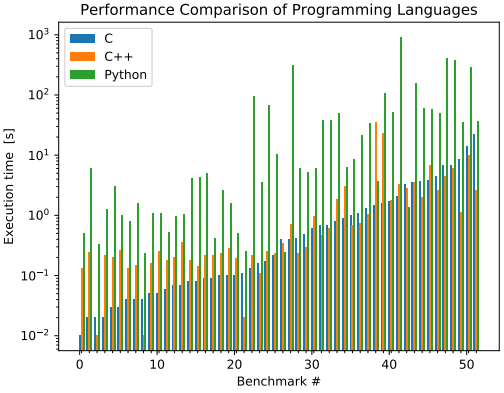
Sixth edition
2020-12

**Programming languages — C++**

*Langages de programmation — C++*

C++ je standardizovan ali (veoma) živ jezik

# Zašto baš C++?




Izvor: https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

C++ je veoma brz

# Zašto baš C++?

A. R. Booker *Res. Number Theory* (2019) 5:26
https://doi.org/10.1007/s40993-019-0162-1

Research in Number Theory

RESEARCH

## Cracking the problem with 33

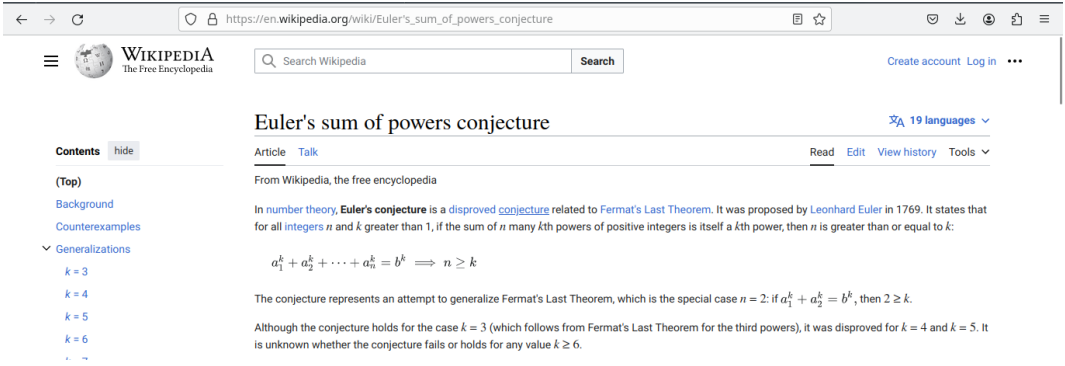
Andrew R. Booker

### 3 Implementation

We implemented the above algorithm in C, with a few inline assembly routines for Montgomery arithmetic [14] written by Buhrow [4], and Walisch's `primesieve` library [19] for enumerating prime numbers.

# Jedan konkretan primer

# Ojlerova hipoteza



# Ojlerova hipoteza



2. F. P. Ramsey, *On a problem of formal logic*, Proc. London Math. Soc. (2) 30 (1930), 264–286.

DARTMOUTH COLLEGE

---

## COUNTEREXAMPLE TO EULER'S CONJECTURE ON SUMS OF LIKE POWERS

BY L. J. LANDER AND T. R. PARKIN

Communicated by J. D. Swift, June 27, 1966

A direct search on the CDC 6600 yielded

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$

as the smallest instance in which four fifth powers sum to a fifth power. This is a counterexample to a conjecture by Euler [1] that at least $n$ $n$th powers are required to sum to an $n$th power, $n > 2$.

### REFERENCE

1. L. E. Dickson, *History of the theory of numbers*, Vol. 2, Chelsea, New York, 1952, p. 648.

**Pretpostavka**
*Za svako $n > 2$, potrebno je najmanje $n$ priordnih brojeva da bi zbir njihovih $n$-tih stepena bio $n$-ti stepen nekog prirodnog broja.*

Kako da nađemo kontraprimer primenom „grube sile”?

# Algoritam

1. Izlistavamo redom (leksikografski) kombinacije $n-1$ brojeva iz $[1, k]$, $k \in \mathbb{N}$, $a_1, a_2, ..., a_{n-1}$
2. Za svaku kombinaciju izračunamo $S = \sum_{i=1}^{n-1} a_i^n$
3. Pretpostavimo da je $S$ oblika $b^n$
4. Izračunamo $b$ kao $\lfloor \sqrt[n]{S} \rfloor$
5. Ako je $S = b^n$, našli smo kontraprimer

# Jedna prosta implementacija u Python-u

```
import math
import itertools

najveca_osnova = 150
izlozilac = 5

for komb in itertools.combinations_with_replacement(range(1, najveca_osnova + 1), izlozilac - 1):
    zbir_stepena = sum([elem ** izlozilac for elem in komb])
    osnova_zbira = int(math.floor(zbir_stepena) ** (1.0 / (izlozilac)))
    if zbir_stepena == osnova_zbira ** izlozilac:
        print("Kontraprimer", tuple(list(komb) + [int(math.floor(osnova_zbira))]))
        break
~
~
~
~
~
~
~
~
~
~
~
~
"euler_sums.py" 12L, 444B written
```

# Jedna prosta implementacija u Python-u

```
Kontraprimer (27, 84, 110, 133, 144)

real    0m44.853s
user    0m44.762s
sys     0m0.022s
```

# Ojlerova hipoteza



2. F. P. Ramsey, *On a problem of formal logic*, Proc. London Math. Soc. (2) 30 (1930), 264–286.

DARTMOUTH COLLEGE

---

## COUNTEREXAMPLE TO EULER'S CONJECTURE ON SUMS OF LIKE POWERS

BY L. J. LANDER AND T. R. PARKIN

Communicated by J. D. Swift, June 27, 1966

A direct search on the CDC 6600 yielded

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$

as the smallest instance in which four fifth powers sum to a fifth power. This is a counterexample to a conjecture by Euler [1] that at least $n$ $n$th powers are required to sum to an $n$th power, $n > 2$.

### REFERENCE

1. L. E. Dickson, *History of the theory of numbers*, Vol. 2, Chelsea, New York, 1952, p. 648.

# Kako bi to izgledalo u C++-u?

```cpp
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

int main()
{
	const unsigned najveca_osnova = 150;
	const unsigned izlozilac = 5;

	vector<unsigned> komb(izlozilac - 1);
	for(unsigned d = 0; d < izlozilac; ++d)
		komb[d] = 1;

	unsigned long long broj_kombinacija = ceil(pow(najveca_osnova, izlozilac - 1));
	for(unsigned long long i = 0; i < broj_kombinacija; ++i)
	{
		unsigned long long zbir_stepena = 0;
		for(unsigned d = 0; d < komb.size(); ++d)
			zbir_stepena += pow(komb[d], izlozilac);

		unsigned osnova_zbira = int(pow(zbir_stepena, 1.0 / izlozilac));
		if(zbir_stepena == pow(osnova_zbira, izlozilac))
		{
			for(unsigned d = 0; d < komb.size(); ++d)
			{
				cout << komb[d] << " ^ " << izlozilac;
				if(d < komb.size() - 1)
					cout << " + ";
			}
			cout << " = " << osnova_zbira << " ^ " << izlozilac << endl;
			break;
		}
		bool prenos = true;
		for(unsigned d = 0; d < komb.size(); ++d)
		{
			if(prenos)
			{
				++komb[d];
				if(komb[d] <= najveca_osnova)
					prenos = false;
				else
					komb[d] = 1;
			}
		}

	}

	return 0;
}
```

# Kako bi to izgledalo u C++-u?

```cpp
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

int main()
{
        const unsigned najveca_osnova = 150;
        const unsigned izlozilac = 5;




        unsigned long long broj_kombinacija = ceil(pow(najveca_osnova, izlozilac - 1));
        for(unsigned long long i = 0; i < broj_kombinacija; ++i)
        {
                unsigned long long zbir_stepena = 0;
                for(unsigned d = 0; d < komb.size(); ++d)
                        zbir_stepena += pow(komb[d], izlozilac);

                unsigned osnova_zbira = int(pow(zbir_stepena, 1.0 / izlozilac));
                if(zbir_stepena == pow(osnova_zbira, izlozilac))
                {









                        break;
```

Suštinski deo je veoma sličan

# Kako bi to izgledalo u C++-u?

Ispis kombinacije je malo složeniji

```cpp
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

int main()
{
        const unsigned najveca_osnova = 150;
        const unsigned izlozilac = 5;




        unsigned long long broj_kombinacija = ceil(pow(najveca_osnova, izlozilac - 1));
        for(unsigned long long i = 0; i < broj_kombinacija; ++i)
        {
                unsigned long long zbir_stepena = 0;
                for(unsigned d = 0; d < komb.size(); ++d)
                        zbir_stepena += pow(komb[d], izlozilac);

                unsigned osnova_zbira = int(pow(zbir_stepena, 1.0 / izlozilac));
                if(zbir_stepena == pow(osnova_zbira, izlozilac))
                {
                        for(unsigned d = 0; d < komb.size(); ++d)
                        {
                                cout << komb[d] << " ^ " << izlozilac;
                                if(d < komb.size() - 1)
                                        cout << " + ";
                        }
                        cout << " = " << osnova_zbira << " ^ " << izlozilac << endl;
                        break;
```

# Kako bi to izgledalo u C++-u?

Kao i njihovo izlistavanje

```cpp
#include <iostream>
#include <cmath>
#include <vector>

using namespace std;

int main()
{
	const unsigned najveca_osnova = 150;
	const unsigned izlozilac = 5;

	vector<unsigned> komb(izlozilac - 1);
	for(unsigned d = 0; d < izlozilac; ++d)
		komb[d] = 1;

	unsigned long long broj_kombinacija = ceil(pow(najveca_osnova, izlozilac - 1));
	for(unsigned long long i = 0; i < broj_kombinacija; ++i)
	{
		unsigned long long zbir_stepena = 0;
		for(unsigned d = 0; d < komb.size(); ++d)
			zbir_stepena += pow(komb[d], izlozilac);

		unsigned osnova_zbira = int(pow(zbir_stepena, 1.0 / izlozilac));
		if(zbir_stepena == pow(osnova_zbira, izlozilac))
		{
			for(unsigned d = 0; d < komb.size(); ++d)
			{
				cout << komb[d] << " ^ " << izlozilac;
				if(d < komb.size() - 1)
					cout << " + ";
			}
			cout << " = " << osnova_zbira << " ^ " << izlozilac << endl;
			break;
		}

		bool prenos = true;
		for(unsigned d = 0; d < komb.size(); ++d)
		{
			if(prenos)
			{
				++komb[d];
				if(komb[d] <= najveca_osnova)
					prenos = false;
				else
					komb[d] = 1;
			}
		}
	}

	return 0;
}
```

# Kako bi to izgledalo u C++-u?


Šta će biti sa preostalim ciframa kada ova sa 9 pređe na 0?

# Kako bi to izgledalo u C++-u?

```
133 ^ 5 + 110 ^ 5 + 84 ^ 5 + 27 ^ 5 = 144 ^ 5

real    0m5.255s
user    0m5.245s
sys     0m0.009s
```

Implementacija u C++-u je oko 9× brža

Možemo li nekako da poboljšamo implementaciju u Python-u?

# Najpre moramo da vidimo gde trošimo najviše vremena

```python
import math
import itertools

najveca_osnova = 150
izlozilac = 5

@profile
def nadji_kontraprimer(najveca_osnova, izlozilac):
    for komb in itertools.combinations_with_replacement(range(1, najveca_osnova + 1), izlozilac - 1):
        zbir_stepena = sum([elem ** izlozilac for elem in komb])
        osnova_zbira = int(math.floor(zbir_stepena) ** (1.0 / (izlozilac)))
        if zbir_stepena == osnova_zbira ** izlozilac:
            print("Kontraprimer", tuple(list(komb) + [int(math.floor(osnova_zbira))]))
            break

nadji_kontraprimer(najveca_osnova, izlozilac)
```

kernprof -l euler_sums.py

# Najpre moramo da vidimo gde trošimo najviše vremena

```
Timer unit: 1e-06 s

Total time: 43.3416 s
File: euler_sums_for_profiling.py
Function: nadji_kontraprimer at line 7

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
     7                                           @profile
     8                                           def nadji_kontraprimer(najveca_osnova, izlozilac):
     9  11887772    3170424.7      0.3      7.3      for komb in itertools.combinations_with_replacement(range(1, najveca_osnova + 1), izlozilac - 1):
    10  11887772   25286121.3      2.1     58.3          zbir_stepena = sum([elem ** izlozilac for elem in komb])
    11  11887772    9661269.9      0.8     22.3          osnova_zbira = int(math.floor(zbir_stepena) ** (1.0 / (izlozilac)))
    12  11887772    5223745.2      0.4     12.1          if zbir_stepena == osnova_zbira ** izlozilac:
    13         1         45.8     45.8      0.0              print("Kontraprimer", tuple(list(komb) + [int(math.floor(osnova_zbira))]))
    14         1          1.6      1.6      0.0              break

 43.34 seconds - euler_sums_for_profiling.py:7 - nadji_kontraprimer
```

92,7% vremena je utrošeno u naredbama koje sadrže računanje petog stepena brojeva < 151

Ima li neko ideju kako da poboljšamo ovo rešenje?

# Lukap tabela

Ideja: Umesto da mnogo puta računamo istu operaciju nad istim operandima, možemo je izračunati samo jednom, sačuvati rezultat i kasnije ga naprosto pročitati kada nam ponovo bude potreban

# Lukap tabela

```
import math
import itertools

najveca_osnova = 150
izlozilac = 5

tabela_stepena = {}
tabela_osnova = {}
najveca_osnova_zbira = int(math.ceil(((najveca_osnova ** izlozilac) * (izlozilac - 1)) ** (1.0 / izlozilac)))
for i in range(1, najveca_osnova_zbira + 1):
    stepen = i ** izlozilac
    tabela_stepena.update({i : stepen})
    tabela_osnova.update({stepen : i})


for komb in itertools.combinations_with_replacement(range(1, najveca_osnova + 1), izlozilac - 1):
    osnova_zbira = tabela_osnova.get(sum([tabela_stepena[elem] for elem in komb]), None)
    if osnova_zbira is not None:
        print("Kontraprimer", tuple(list(komb) + [int(math.floor(osnova_zbira))]))
        break
~
~
"euler_sums_with_precomp.py" 20L, 686B                                  16,51         All
```

# Lukap tabela

```
Kontraprimer (27, 84, 110, 133, 144)

real    0m14.022s
user    0m14.004s
sys     0m0.009s
```

# Lukap tabela

Naravno, lukap tabelu možemo iskoristiti da poboljšamo i C++ implementaciju

# Lukap tabela u C++-u

```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <map>

using namespace std;

int main()
{
	const unsigned najveca_osnova = 150;
	const unsigned izlozilac = 5;

	vector<unsigned long long> tabela_stepena(najveca_osnova + 1);
	map<unsigned long long, unsigned> tabela_osnova;
	for(unsigned i = 0; i <= najveca_osnova; ++i)
	{
		unsigned long long stepen = pow(i, izlozilac);
		tabela_stepena[i] = stepen;
		tabela_osnova[stepen] = i;
	}

	vector<unsigned> komb(izlozilac - 1);
	for(unsigned d = 0; d < izlozilac; ++d)
		komb[d] = 1;

	unsigned long long broj_kombinacija = ceil(pow(najveca_osnova, izlozilac - 1));
	for(unsigned long long i = 0; i < broj_kombinacija; ++i)
	{
		unsigned long long zbir_stepena = 0;
		for(unsigned d = 0; d < komb.size(); ++d)
			zbir_stepena += tabela_stepena[komb[d]];

		auto osnova_zbira = tabela_osnova.find(zbir_stepena);
		if(osnova_zbira != tabela_osnova.end())
		{
			for(unsigned d = 0; d < komb.size(); ++d)
			{
				cout << komb[d] << " ^ " << izlozilac;
				if(d < komb.size() - 1)
					cout << " + ";
			}
			cout << " = " << osnova_zbira -> second << " ^ " << izlozilac << endl;
			break;
		}

		bool prenos = true;
		for(unsigned d = 0; d < komb.size(); ++d)
		{
			if(prenos)
			{
				++komb[d];
				if(komb[d] <= najveca_osnova)
					prenos = false;
				else
					komb[d] = 1;
			}
		}
	}

	return 0;
}
```

# Lukap tabela u C++-u

```
133 ^ 5 + 110 ^ 5 + 84 ^ 5 + 27 ^ 5 = 144 ^ 5

real    0m2.165s
user    0m2.150s
sys     0m0.012s
```

# Preporuka

- Početi od najjednostavnije implementacije (na primer u Python-u)
- Ako se pokaže da za dati problem nije dovoljno efikasna, tek onda preći na složeniju (na primer u C++-u)
- Često se daleko više isplati uložiti vreme u osmišljavanje boljeg algoritma nego u pisanje bolje implementacije

No, često je i ubrzanje od (nekoliko) red(ova) veličine koje se može postići efikasnijom implementacijom presudno za uspešno rešavanje posebno teških instanci nekog problema

Cilj ovog predmeta je da se upoznamo sa programskim jezikom C++, pa izbor jezika uglavnom neće biti slobodan, ali ćemo na kraju semestra ipak analizirati mogućnosti integracije programskog koda napisanog, spram potreba, u Python-u, C++-u i asembliju

# Zaključci

- Računari mogu biti od velike pomoći u matematici (za formulisanje hipoteza, njihovo obaranje, pronalaženje ključnih elemenata dokaza...)

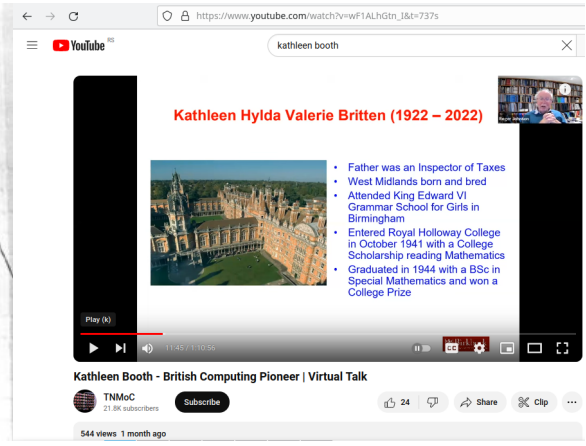
# Zaključci

- Računari mogu biti od velike pomoći u matematici (za formulisanje hipoteza, njihovo obaranje, pronalaženje ključnih elemenata dokaza...)
- Matematičari su dali nemerljiv doprinos uspostavljanju i razvoju računarstva kao zasebne naučne oblasti

# Zaključci

- Računari mogu biti od velike pomoći u matematici (za formulisanje hipoteza, njihovo obaranje, pronalaženje ključnih elemenata dokaza...)
- Matematičari su dali nemerljiv doprinos uspostavljanju i razvoju računarstva kao zasebne naučne oblasti
- To se odnosi kako na matematičare posvećene nauci, tako i na profesore matematike u srednjim školama

Ketlin But


https://www.youtube.com/watch?v=wF1ALhGtn_I&t=737s
YouTube
kathleen booth
Kathleen Hylda Valerie Britten (1922 – 2022)
Father was an Inspector of Taxes
West Midlands born and bred
Attended King Edward VI Grammar School for Girls in Birmingham
Entered Royal Holloway College in October 1941 with a College Scholarship reading Mathematics
Graduated in 1944 with a BSc in Special Mathematics and won a College Prize
Play (k)
11:45 / 1:10:56
Kathleen Booth - British Computing Pioneer | Virtual Talk
TNMoC
21.8K subscribers
Subscribe
24
Share
Clip
544 views 1 month ago


- Pamtite li neke profesore koji su na vas izvršili značajan uticaj?

- Ne baš. Čini mi se da sam u tom trenutku, na fakultetu, bio poprilično samostalan. Možda nisam bio potpuno formiran, ali sigurno nisam bio ni u formativnim godinama. Takođe mislim da, kada je reč o nastavnicima, pored mog dede, najuticajniji u tom smislu je bio moj nastavnik matematike u srednjoj školi.

- S kojim nivoom matematike vas je upoznao? Sa infinitezimalnim računom?

- Matematika... Mislim da je matematika u finskim srednjim školama već na višem nivou nego što je verovatno ovde, osim ako neko ne izabere neke napredne matematičke predmete u SAD-u. No, da to navedem samo kao primer, taj profesor je napisao udžbenik koji je korišćen i u svim drugim školama. Prvog časa koji smo imali sa njim, rekao je: „Dobro, ja sam napisao ovaj udžbenik, ali ga neću koristiti." Onda se naprosto ponašao na način na koji su nam kasnije predavali matematiku na fakultetu; davao nam je svoje beleške i nismo morali sami da ih hvatamo, jer je smatrao da je to gubljenje i našeg i njegovog vremena. Bilo je mnogo bolje što je svima podelio beleške, pa umesto da pokušavamo da ih zapišemo, trebalo je da slušamo i razmišljamo o tome.

Zapravo, kada sam upisivao fakultet, dvoumio sam se da li da idem na matematiku ili računarstvo, jer sam stvarno mislio da više volim matematiku.

# Zaključci

- Računari mogu biti od velike pomoći u matematici (za formulisanje hipoteza, njihovo obaranje, pronalaženje ključnih elemenata dokaza...)
- Matematičari su dali nemerljiv doprinos uspostavljanju i razvoju računarstva kao zasebne naučne oblasti

Nema nikakvog razloga da i ubuduće ne bude tako

Hvala na pažnji