Такмичење из програмирања за ученике основих школа

Душан Попадић Јелена Петровић Огњен Тешић

4. фебруар 2024.

План за данас

- Циљеви и организација такмичења
- Општа упутства за ученике
- Основни задаци
- Увод у сложеност и типични задаци (сортирање, бинарна претрага, два показивача)
- Задаци из теорије бројева
- Похлепни приступ решавању задатака
- Динамичко програмирање

Циљеви такмичења

- Заинтересовати ученике за програмирање
- Развити алгоритамски начин размишљања
- Подстаћи на самосталан рад
- Развити такмичарски дух
- Показати да се труд исплати
- Олакшати даље образовање
- Избор екипе за међународна такмичења

Нивои такмичења

Ниво	Датум
Квалификације 1. круг	4.11.2023. (10-22)
Квалификације 2. круг	16.12.2023. (10-22)
Општинско такмичење	28.1.2024. (10-12)
Окружно такмичење	2.3.2024. (10-12.30)
Државно такмичење	13.4.2024. (13-16)
Српска информатичка олимпијада	26.5.2024. (13-16.30)
Изборно такмичење	8.6.2024.
Европска јуниорска информатичка олимпијада (Кишињев)	16-23.8.2024.
Балканска јуниорска информатичка олимпијада (Кипар)	24-27.11.2024.

Подржани програмски језици

Програмски језик	Развојно окружење	Опис окружења
C, C++	Code::Blocks	Code::Blocks 17.12 са GCC 5.1.0+
Python	IDLE	Python 3.6+
C#	Visual Studio	MVS 2017

На ЈСИО су омогућени језици C++ и Пајтон, са тим што се не гарантује исти број поена за иста решења у овим језицима. На изборном се омогућава само C++.

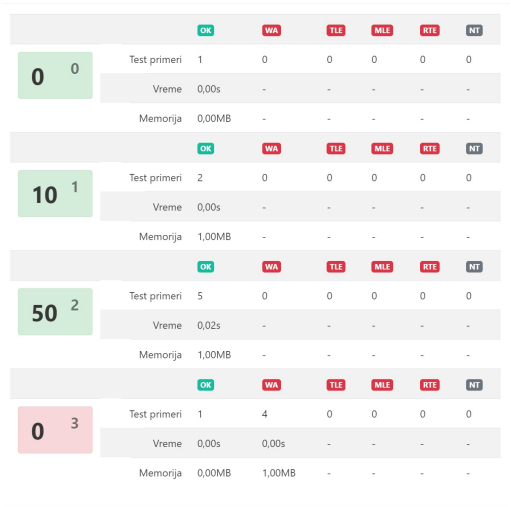
Бодовање

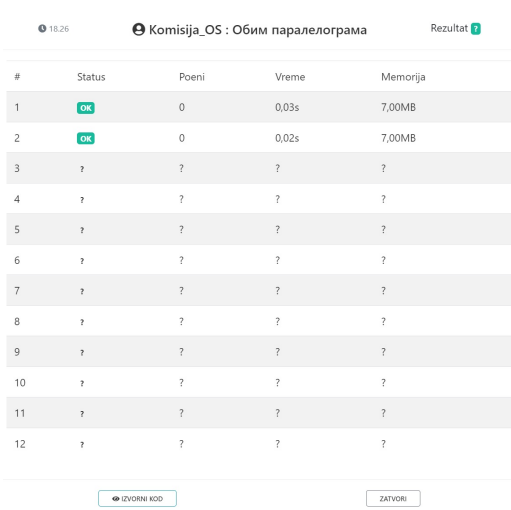
- 3-5 задатака - сваки по 100 поена
- Бодује се по тест примерима или групама тест примера
- Време за рад је 2-3 сата, зависно од нивоа такмичења
- Временска ефикасност

Тест примери

- Да ли су тест примери јавни?
- Чему служе групе тест примера?
- Да ли се оцењује код?

#	Status	Poeni	Vreme	Memorija
1	OK	0	0,00s	0,00MB
2	OK	0	0,02s	1,00MB
3	OK	10	0,00s	2,00MB
4	WA	-	0,00s	0,00MB
5	WA	-	0,02s	1,00MB
6	WA	-	0,00s	0,00MB
7	OK	10	0,16s	4,00MB
8	OK	10	0,16s	5,00MB
9	OK	10	0,19s	5,00MB
10	OK	10	0,17s	4,00MB
11	OK	10	0,19s	5,00MB
12	OK	10	0,17s	5,00MB





Процес жалби

- Жалбе се подносе државној комисији у року који се објављује са прелиминарним резултатима
- „Комисија не може да усвоји жалбу у којој се тражи исправљање грешке у алгоритму, ма колико мала она била. Једине дозвољене исправке у изворном коду такмичара односе се на формат учитавања података и исписивања резултата на Општинском и Окружном нивоу”
- Рок за објављивање коначних резултата је 10 дана

Пример усвојене жалбе (1/2)

Ученик шестог разреда Вук је добио 0 поена за задатак преклапање цифара. У разговору са учеником и увидом у његов задатак када се тестира наведени задатак и унесе улази као што сте дали добија исте излазе као ваше. Решење има недостатака у наведеном задатку за које пишемо приговор, али ипак се ученик трудио и покушавао да реши задатак, тако да сматрам да би требало да добије одређен број поена. Молим вас да опет погледајте овај задатак и уколико сматрате и ви да има елемената за признавање то учините. Корисничко име ученика је: XXXXXXXXXXX

Пример усвојене жалбе (1/2)

Ученик шестог разреда Вук је добио 0 поена за задатак преклапање цифара. У разговору са учеником и увидом у његов задатак када се тестира наведени задатак и унесе улази као што сте дали добија исте излазе као ваше. Решење има недостатака у наведеном задатку за које пишемо приговор, али ипак се ученик трудио и покушавао да реши задатак, тако да сматрам да би требало да добије одређен број поена. Молим вас да опет погледајте овај задатак и уколико сматрате и ви да има елемената за признавање то учините. Корисничко име ученика је: XXXXXXXXXXX
Поштовани, жалба се усваја јер је проблем само у начину исписивања резултата (требало је исписати цифре без размака између њих). Молимо Вас да Вуку скренете пажњу на то да се овакве жалбе не могу усвајати на вишим нивоима такмичења (члан 8.3 правилника).

Пример усвојене жалбе (2/2)

Маре и Паја играју игрицу и дошли су до нивоа где треба решити шифру. Схватили су да је шифра заправо један петоцифрен број који је био записан на зиду на претходном нивоу игрице. Ниједан од њих не може да се сети тог броја, али се сећају делова. Маре је запамтио прве три цифре, а Паја последње три. Помозите им да пређу на следећи ниво тако што ћете за њих одредити тражени петоцифрен број. Претпоставити да је свако од њих исправно запамтио цифре, односно да су унети подаци исправни.

Улаз

У првом реду се налази троцифрен број - део који је запамтио Маре. У другом реду се налази троцифрен број - део који је запамтио Паја.

Излаз

Исписати тражени петоцифрен број.

Пример 1

Улаз

```
145
563
```

Излаз

```
14563
```

Пример усвојене жалбе (2/2)

Маре и Паја играју игрицу и дошли су до нивоа где треба решити шифру. Схватили су да је шифра заправо један петоцифрен број који је био записан на зиду на претходном нивоу игрице. Ниједан од њих не може да се сети тог броја, али се сећају делова. Маре је запамтио прве три цифре, а Паја последње три. Помозите им да пређу на следећи ниво тако што ћете за њих одредити тражени петоцифрен број. Претпоставити да је свако од њих исправно запамтио цифре, односно да су унети подаци исправни.

Улаз

У првом реду се налази троцифрен број - део који је запамтио Маре. У другом реду се налази троцифрен број - део који је запамтио Паја.

Излаз

Исписати тражени петоцифрен број.

Пример 1

Улаз

```
145
563
```

Излаз

```
14563
```

```
m=int(input())
p=int(input())
a=m//100
b=m%100//10
c=m%100%10
d=p//100
```

```
e=p%100//10
f=p%100%10
if c==d:
  print(a,b,c,e,f)
```

```
print(a,b,c,e,f, sep='')
```

Примери одбијених жалби

- Ученик је ставио $\leq$ уместо $<$
- Ученик је написао print(greska) уместо print('greska')
- Ученик је заборавио да пре циклуса стави zbir=0;
- Задатак: Имамо n топова на шаховској табли, исписати да ли бар један напада унето поље. Ученица је независно проверавала по врстама и колонама и онда два пута исписивала „napada" ако је поље нападнуто на оба начина
- Ученик је у угњежђеној петљи бројао нешто, али је brojac=0 ставио ван спољне петље уместо у спољној петљи

Регуларност такмичења

- Дежурни наставници
- Код за приступ такмичењу
- Провера сличности кодова
- Провера ИП адреса
- Аутоматски решавачи задатака?

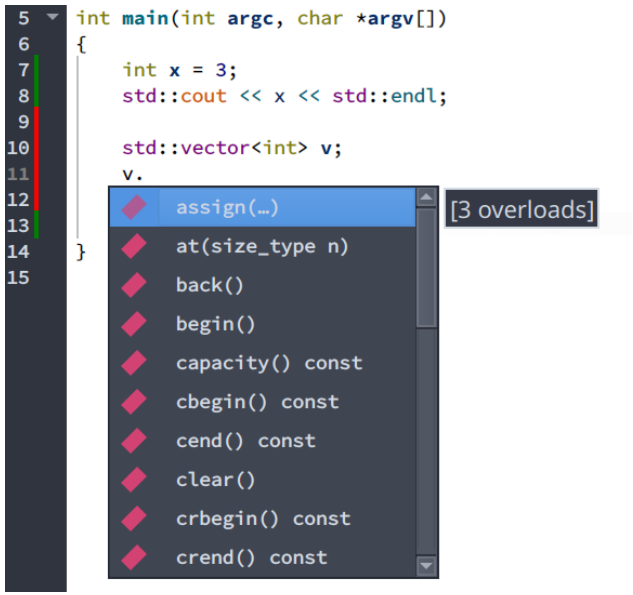
Често постављана питања

- Да ли треба да проверавамо услове за улазне податке? Не.
- Могу ли да се такмичим ако идем у пети разред? Да.
- Чуо/чула сам да је аутоматско прегледање решења прилично строго. О чему се ради?
- Ако на тестовима из поставке задатка добијем статус „OK” да ли то значи да ми је решење тачно?
- Зашто не видим резултате аутоматског тестирања док траје такмичење?
- Ако се задаци аутоматски прегледају зашто комисији треба некад и 2 дана да објави прелиминарне резултате?
- Зашто се одузимају поени за решења која за делић секунде прекораче време?

https://takprog.petlja.org/osnovnaskola/posts/17

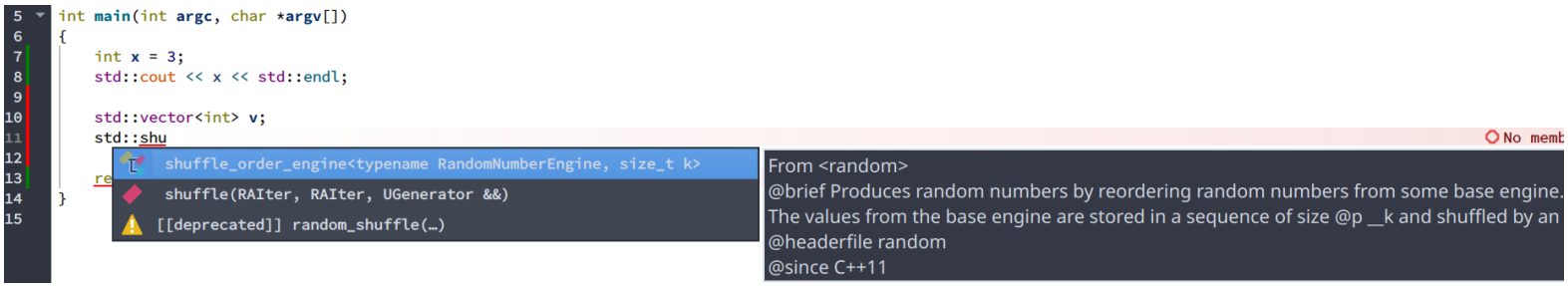
Познавање окружења

- Доступне функције и потписи



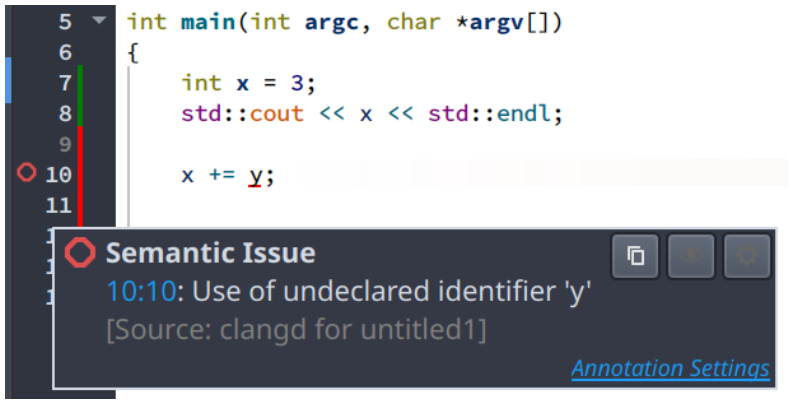
Познавање окружења

- Доступне функције и потписи
- Описи функција



Познавање окружења

- Доступне функције и потписи
- Описи функција
- Грешке и упозорења



Познавање окружења

- Доступне функције и потписи
- Описи функција
- Грешке и упозорења
- Форматирање кода, бојење синтаксе

Познавање окружења

- Доступне функције и потписи
- Описи функција
- Грешке и упозорења
- Форматирање кода, бојење синтаксе
- Алати

Познавање окружења

- Доступне функције и потписи
- Описи функција
- Грешке и упозорења
- Форматирање кода, бојење синтаксе
- Алати
- Пречице

Форматирање и структура кода

- Смислени називи ф-ја и променљивих, белине, заграде

Форматирање и структура кода

- Смислени називи ф-ја и променљивих, белине, заграде
- Рашчлањивање програма на ф-је, коришћење структура

Форматирање и структура кода

- Смислени називи ф-ја и променљивих, белине, заграде
- Рашчлањивање програма на ф-је, коришћење структура
- Избегавање компликованих аспеката језика

Форматирање и структура кода

- Смислени називи ф-ја и променљивих, белине, заграде
- Рашчлањивање програма на ф-је, коришћење структура
- Избегавање компликованих аспеката језика

```
int f(int n) {
int ooo = 0, oooo = 1;
int c;
for(c=2; c<=n; ++c){
int ooooo = ooo + oooo;
ooo = oooo;
oooo = ooooo;
}return oooo;}
```

```
int fibonaci(int n) {
    int prethodni = 0;
    int trenutni = 1;

    for (int i = 2; i <= n; ++i) {
        int sledeci = prethodni +
        ↪  trenutni;
        prethodni = trenutni;
        trenutni = sledeci;
    }

    return trenutni;
}
```

Стандардна библиотека

- Олакшава и убрзава рад, умањује могућност грешке

Стандардна библиотека

- Олакшава и убрзава рад, умањује могућност грешке
- **Колекције**: вектор, мапа, ред, скуп, пар, стринг

```
vector<int> vektor{1, 2, 3};
map<string, int> mapa{{"Vuk", 10}, {"Ana", 8}};
string niska = "Pera";
```

Стандардна библиотека

- Олакшава и убрзава рад, умањује могућност грешке
- **Колекције**: вектор, мапа, ред, скуп, пар, стринг

```
vector<int> vektor{1, 2, 3};
map<string, int> mapa{{"Vuk", 10}, {"Ana", 8}};
string niska = "Pera";
```

- Функције за рад са колекцијама, математичке функције

```
string podniska = niska.substr(0, 3);
float x = log(10);
```

Стандардна библиотека

- Олакшава и убрзава рад, умањује могућност грешке
- **Колекције**: вектор, мапа, ред, скуп, пар, стринг

```
vector<int> vektor{1, 2, 3};
map<string, int> mapa{{"Vuk", 10}, {"Ana", 8}};
string niska = "Pera";
```

- Функције за рад са колекцијама, математичке функције

```
string podniska = niska.substr(0, 3);
float x = log(10);
```

- **Алгоритми**: сортирање, бинарна претрага

```
sort(vektor.begin(), vektor.end());
binary_search(vektor.begin(), vektor.end(), broj);
```

Стандардна библиотека

- Олакшава и убрзава рад, умањује могућност грешке
- **Колекције**: вектор, мапа, ред, скуп, пар, стринг

```
vector<int> vektor{1, 2, 3};
map<string, int> mapa{{"Vuk", 10}, {"Ana", 8}};
string niska = "Pera";
```

- Функције за рад са колекцијама, математичке функције

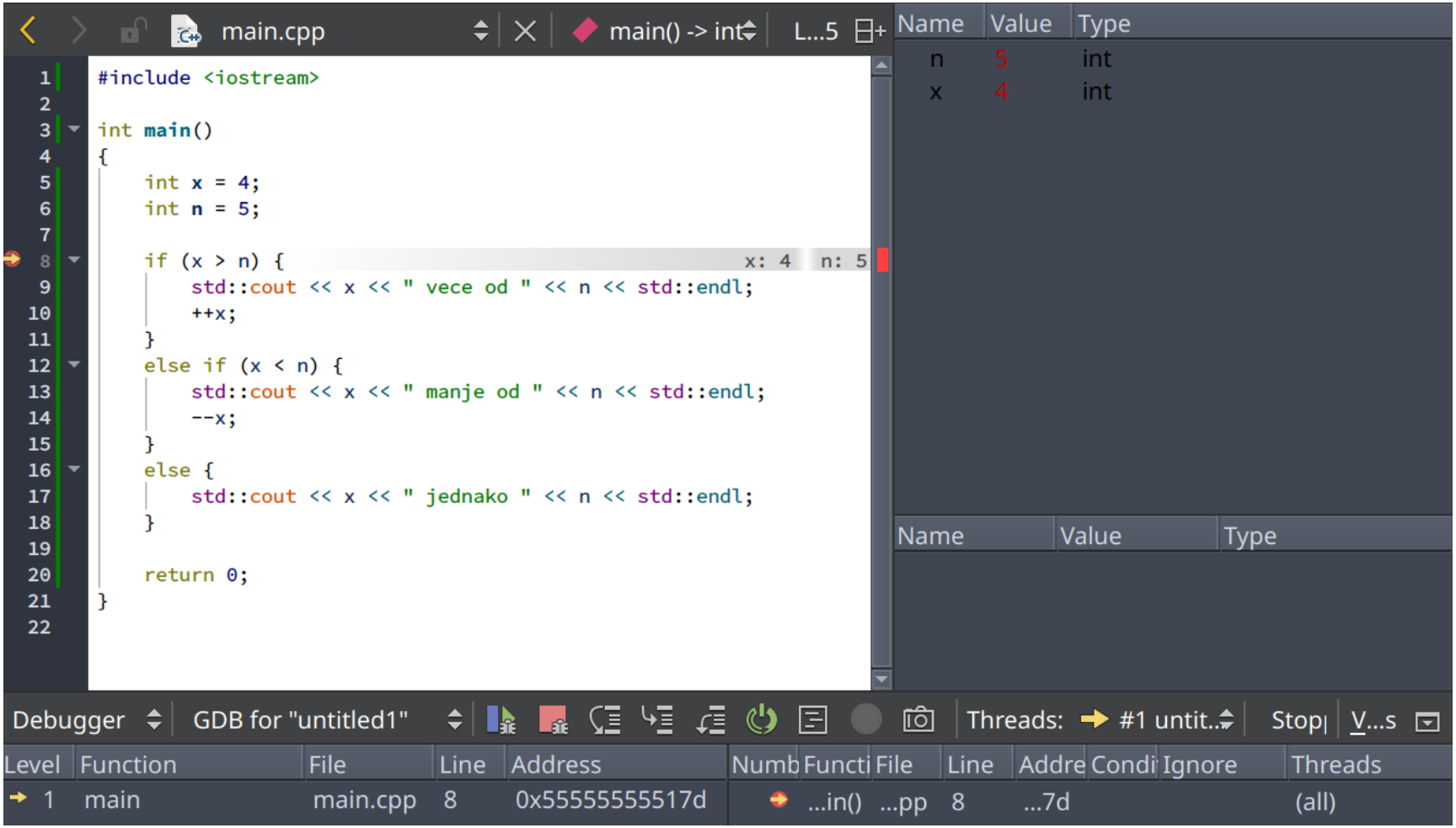
```
string podniska = niska.substr(0, 3);
float x = log(10);
```

- **Алгоритми**: сортирање, бинарна претрага

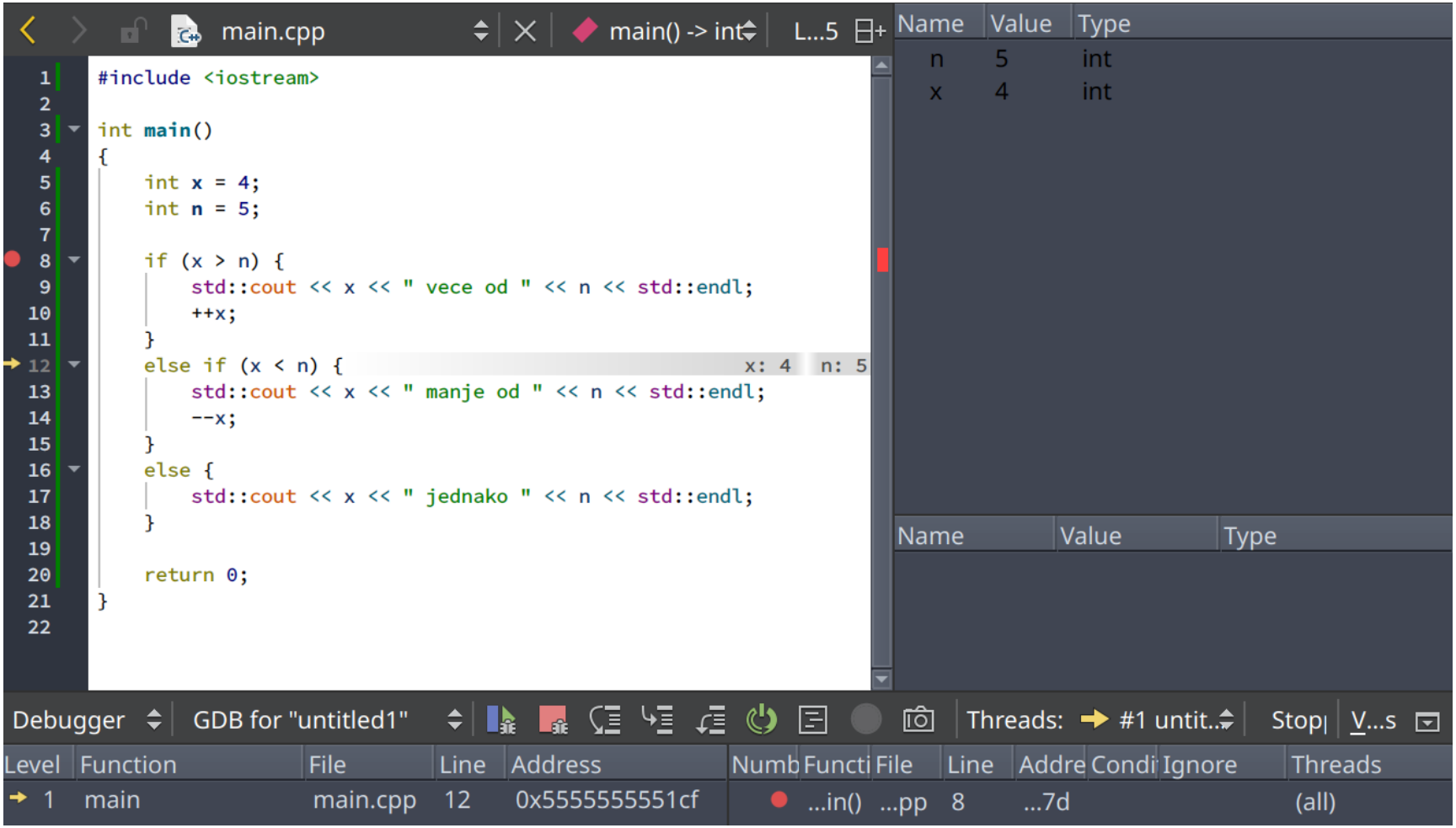
```
sort(vektor.begin(), vektor.end());
binary_search(vektor.begin(), vektor.end(), broj);
```

- **Константе**: π, e

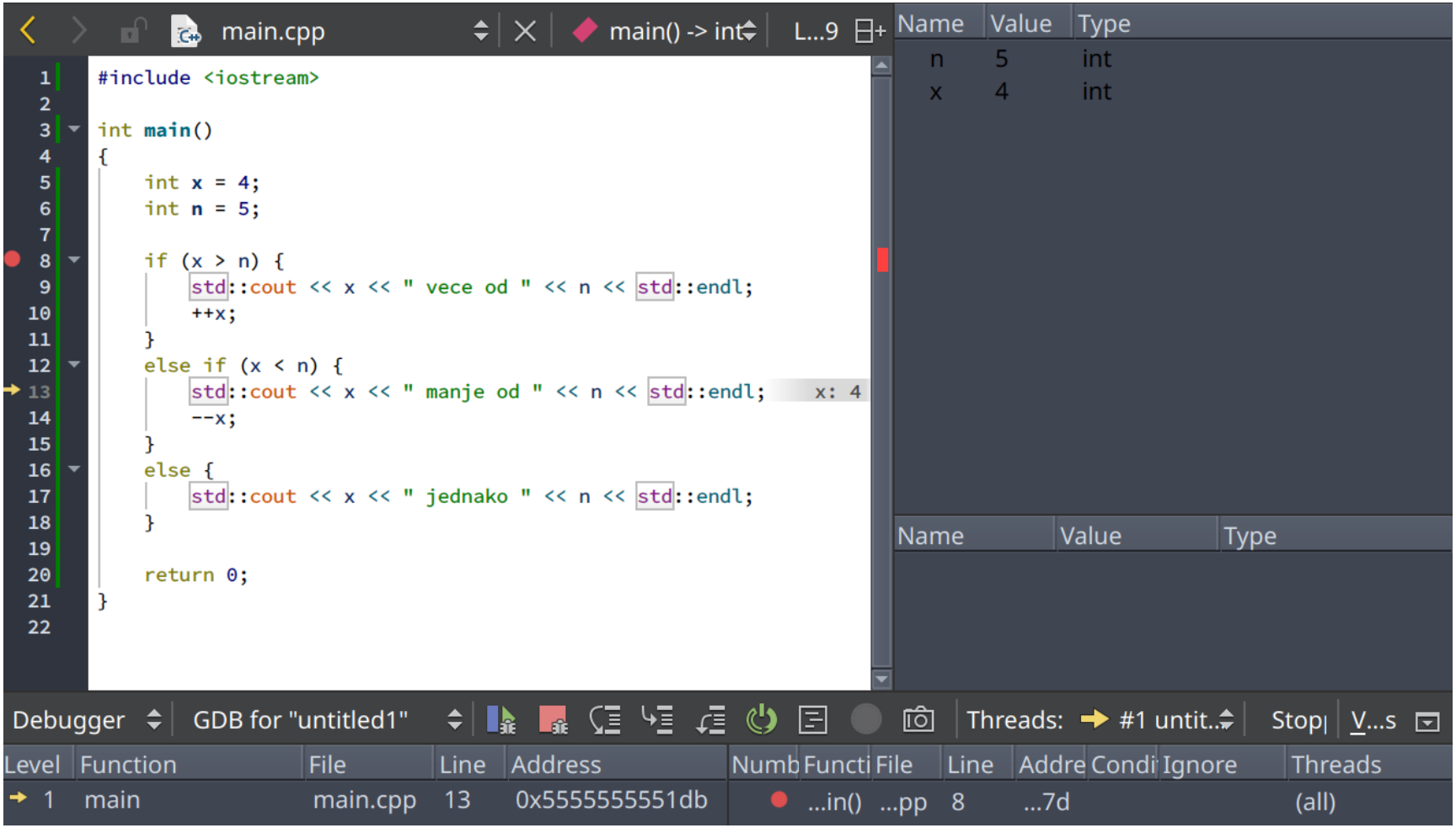
Дебаговање


main.cpp
main() -> int
L...5
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
x: 4
n: 5
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
std::cout << x << " manje od " << n << std::endl;
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
}
return 0;
}
Name
Value
Type
n
5
int
x
4
int
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
V...s
Level
Function
File
Line
Address
1
main
main.cpp
8
0x55555555517d
...in()
...pp
8
...7d
(all)

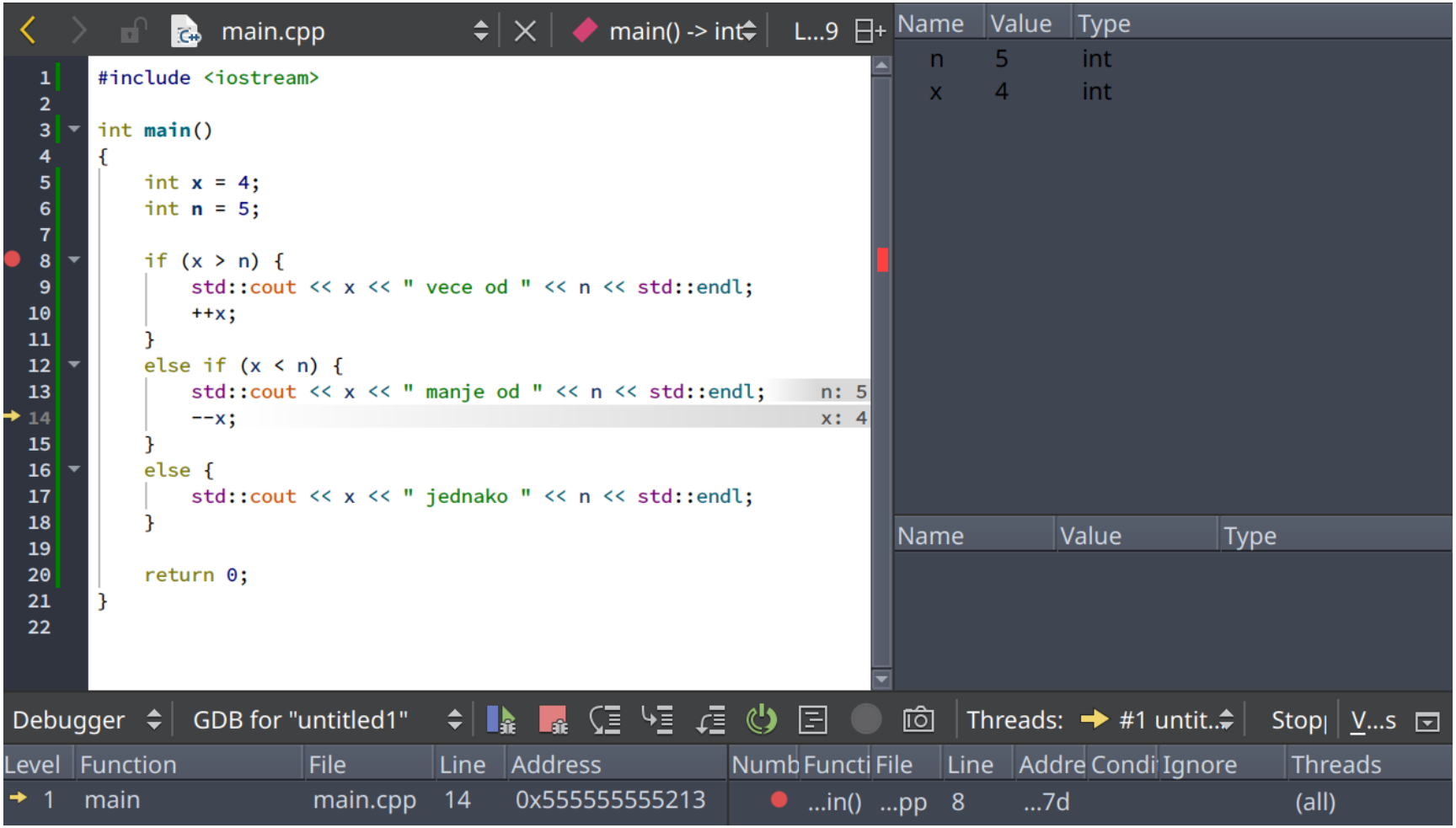
Дебаговање


main.cpp
main() -> int
L...5
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
x: 4 n: 5
std::cout << x << " manje od " << n << std::endl;
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
}
return 0;
}
Name Value Type
n 5 int
x 4 int
Name Value Type
Debugger
GDB for "untitled1"
Threads: #1 untit...
Stop
V...s
Level Function File Line Address
1 main main.cpp 12 0x5555555551cf
Numb Functi File Line Addre Condi Ignore Threads
...in() ...pp 8 ...7d (all)

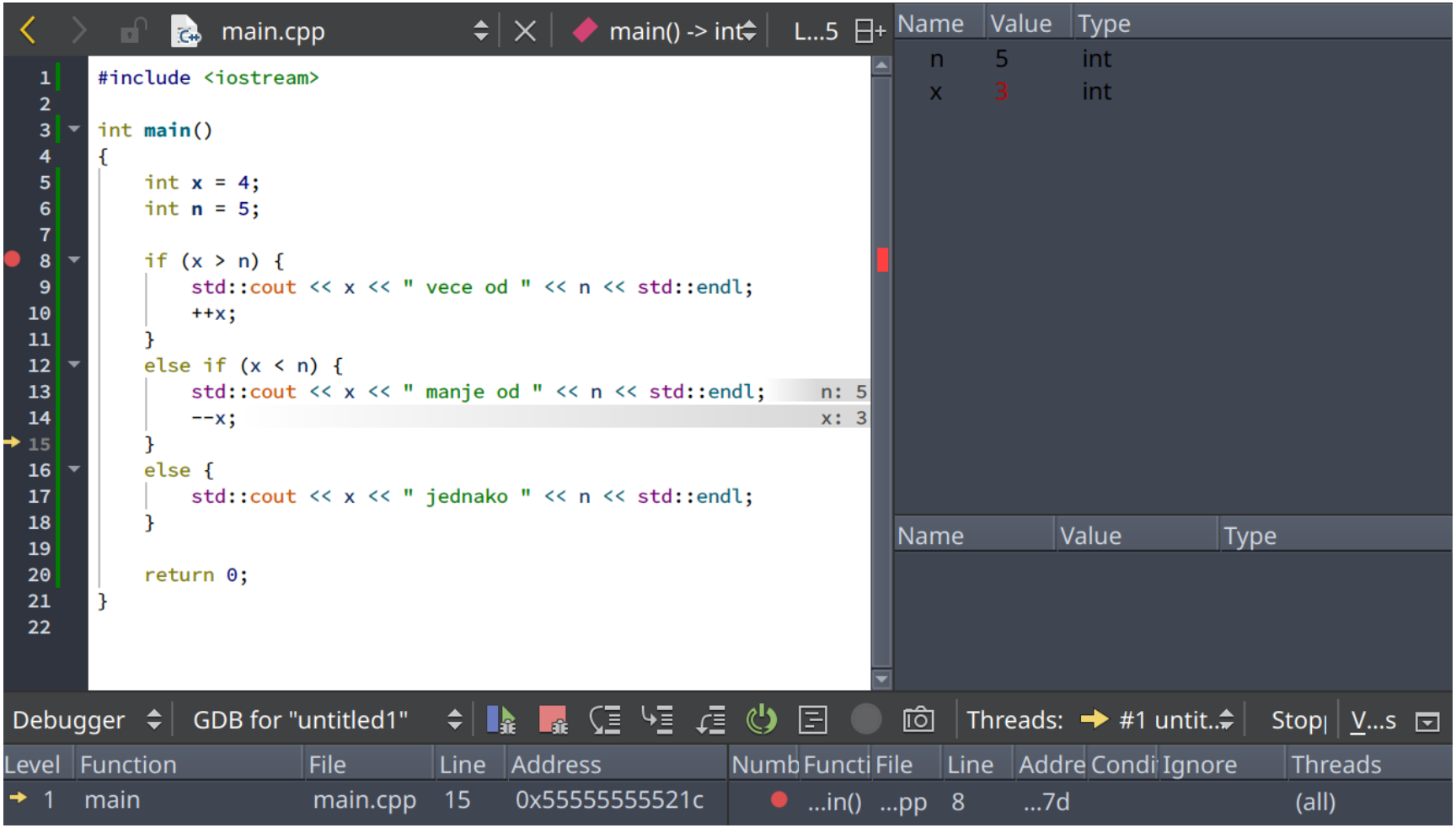
Дебаговање


main.cpp
main() -> int
L...9
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
std::cout << x << " manje od " << n << std::endl;
x: 4
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
}
return 0;
}
Name Value Type
n 5 int
x 4 int
Name Value Type
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
V...s
Level Function File Line Address
1 main main.cpp 13 0x5555555551db
Numb Functi File Line Addre Condi Ignore Threads
...in() ...pp 8 ...7d (all)

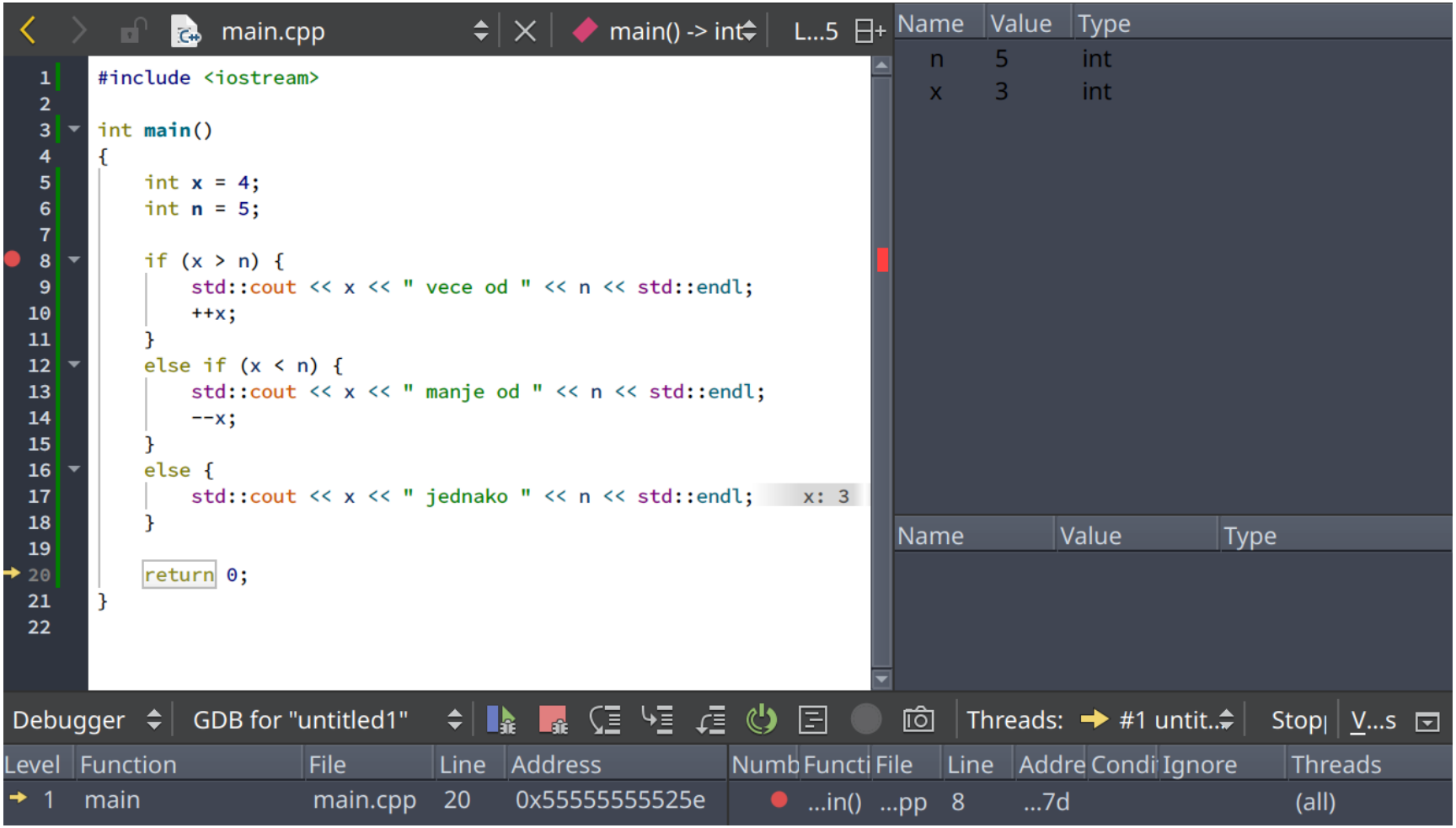
Дебаговање


main.cpp
main() -> int
L...9
1 #include <iostream>
2
3 int main()
4 {
5 int x = 4;
6 int n = 5;
7
8 if (x > n) {
9 std::cout << x << " vece od " << n << std::endl;
10 ++x;
11 }
12 else if (x < n) {
13 std::cout << x << " manje od " << n << std::endl;
14 --x;
15 }
16 else {
17 std::cout << x << " jednako " << n << std::endl;
18 }
19
20 return 0;
21 }
22
n: 5
x: 4
Name Value Type
n 5 int
x 4 int
Name Value Type
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
V...s
Level Function File Line Address
1 main main.cpp 14 0x555555555213
Numb Functi File Line Addre Condi Ignore Threads
...in() ...pp 8 ...7d (all)

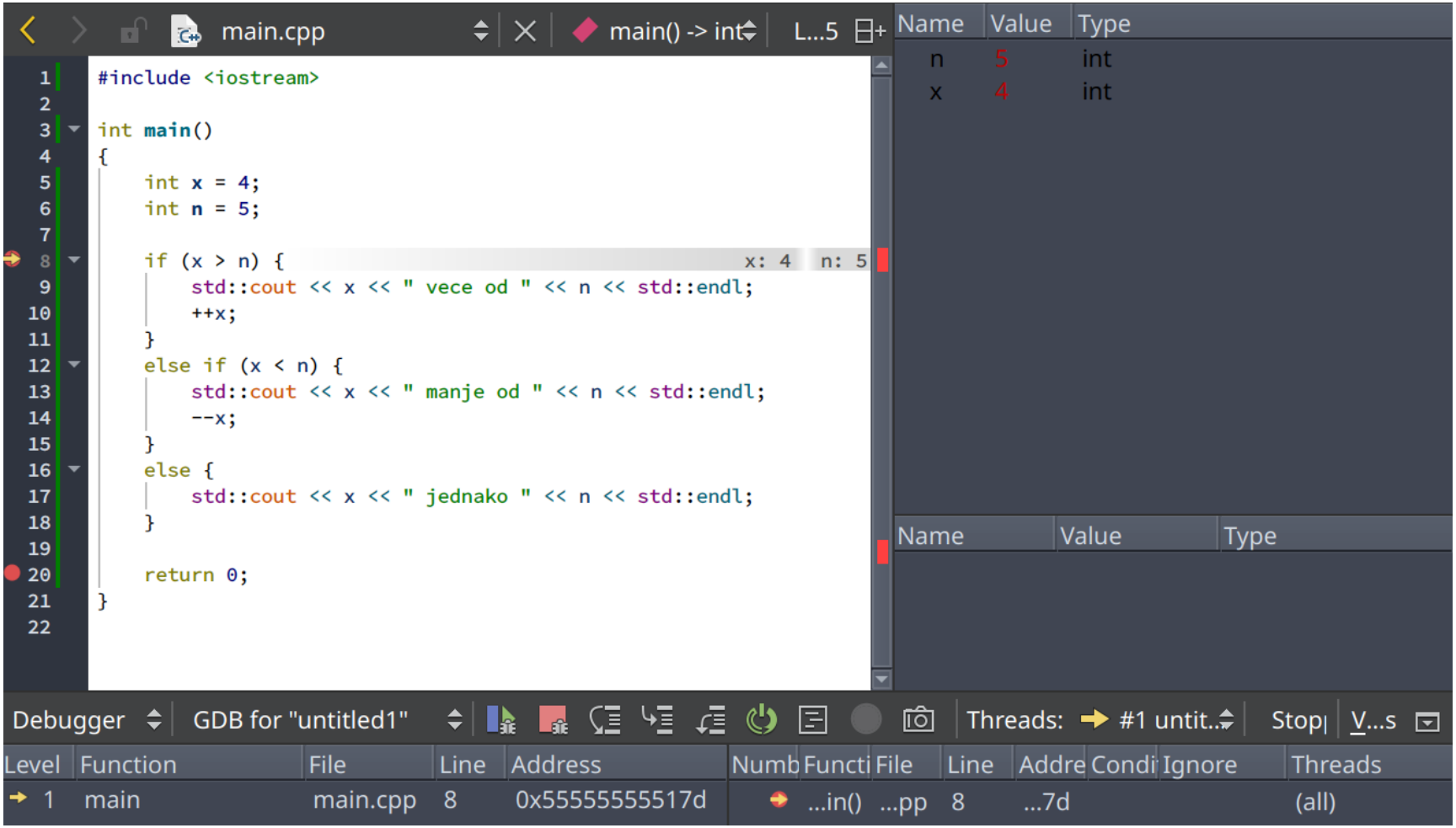
Дебаговање


main.cpp
main() -> int
L...5
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
std::cout << x << " manje od " << n << std::endl;
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
}
return 0;
}
n: 5
x: 3
Name Value Type
n 5 int
x 3 int
Name Value Type
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
V...s
Level Function File Line Address
1 main main.cpp 15 0x55555555521c
Numb Functi File Line Addre Condi Ignore Threads
...in() ...pp 8 ...7d (all)

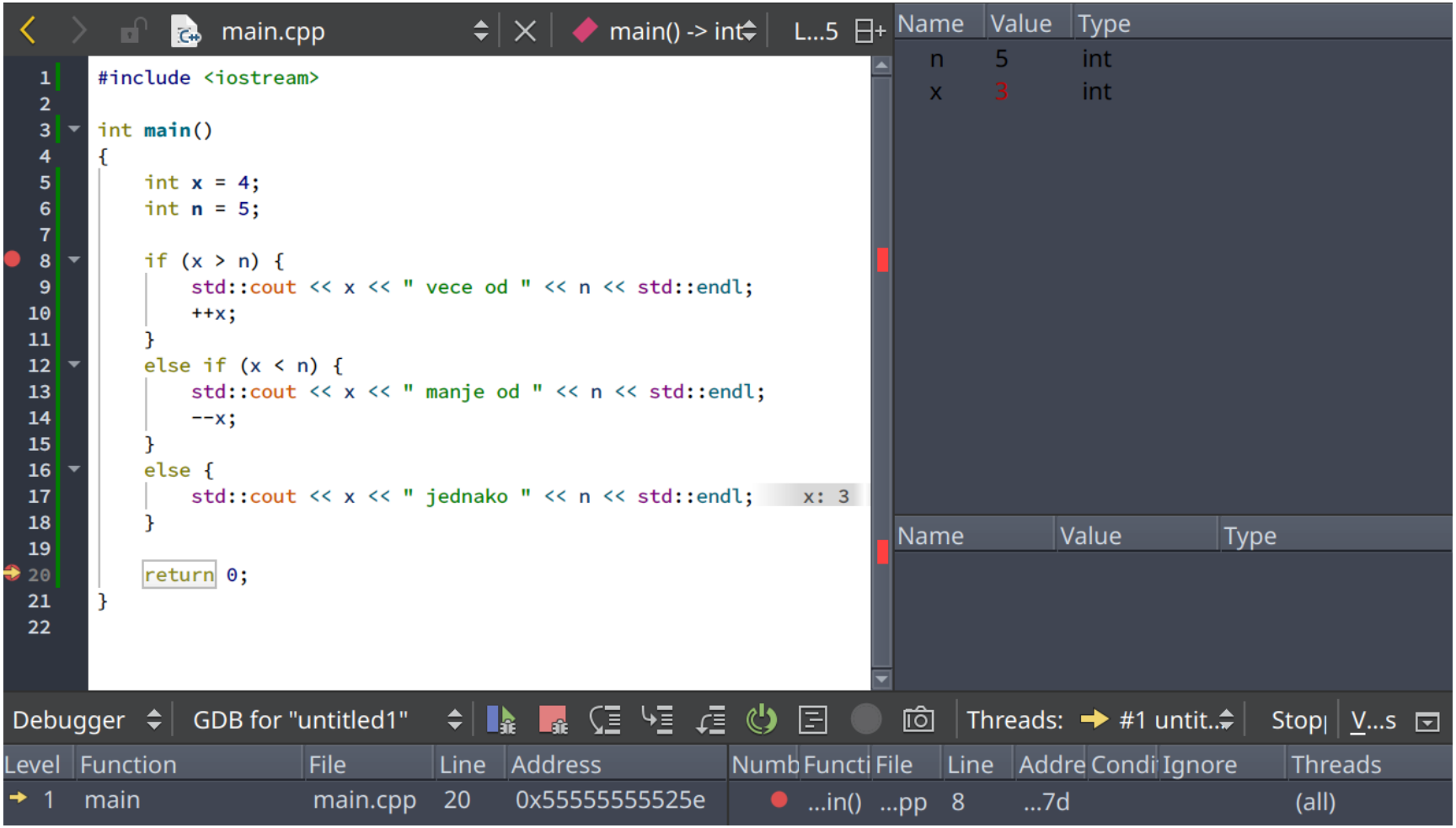
Дебаговање


main.cpp
main() -> int
L...5
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
std::cout << x << " manje od " << n << std::endl;
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
x: 3
}
return 0;
}
Name Value Type
n 5 int
x 3 int
Name Value Type
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
V...s
Level Function File Line Address
1 main main.cpp 20 0x55555555525e
Numb Functi File Line Addre Condi Ignore Threads
...in() ...pp 8 ...7d (all)

Дебаговање


main.cpp
main() -> int
L...5
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
x: 4
n: 5
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
std::cout << x << " manje od " << n << std::endl;
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
}
return 0;
}
Name
Value
Type
n
5
int
x
4
int
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
Level
Function
File
Line
Address
1
main
main.cpp
8
0x55555555517d
...in()
...pp
8
...7d
(all)

Дебаговање


main.cpp
main() -> int
L...5
#include <iostream>
int main()
{
int x = 4;
int n = 5;
if (x > n) {
std::cout << x << " vece od " << n << std::endl;
++x;
}
else if (x < n) {
std::cout << x << " manje od " << n << std::endl;
--x;
}
else {
std::cout << x << " jednako " << n << std::endl;
x: 3
}
return 0;
}
Name Value Type
n 5 int
x 3 int
Name Value Type
Debugger
GDB for "untitled1"
Threads:
#1 untit...
Stop
V...s
Level Function File Line Address
1 main main.cpp 20 0x55555555525e
Numb Functi File Line Addre Condi Ignore Threads
...in() ...pp 8 ...7d (all)

Општи приступ решавању проблема

- Шта је дато, шта се тражи?

Општи приступ решавању проблема

- Шта је дато, шта се тражи?
- Ручно решити задатак на неком примеру и описати кораке

Општи приступ решавању проблема

- Шта је дато, шта се тражи?
- Ручно решити задатак на неком примеру и описати кораке
- Да ли постоје специјални случајеви?

Општи приступ решавању проблема

- Шта је дато, шта се тражи?
- Ручно решити задатак на неком примеру и описати кораке
- Да ли постоје специјални случајеви?
- Да ли може боље?

Општи приступ решавању проблема

- Шта је дато, шта се тражи?
- Ручно решити задатак на неком примеру и описати кораке
- Да ли постоје специјални случајеви?
- Да ли може боље?
- Кораке претворити у наредбе програмског језика

Општи приступ решавању проблема

- Шта је дато, шта се тражи?
- Ручно решити задатак на неком примеру и описати кораке
- Да ли постоје специјални случајеви?
- Да ли може боље?
- Кораке претворити у наредбе програмског језика
- Имплементирати корак по корак уз провере на примерима

Општи приступ решавању проблема

- Шта је дато, шта се тражи?
- Ручно решити задатак на неком примеру и описати кораке
- Да ли постоје специјални случајеви?
- Да ли може боље?
- Кораке претворити у наредбе програмског језика
- Имплементирати корак по корак уз провере на примерима
- Проћи дебагером кроз код

Неке теме предвиђене за Општинско такмичење

- Целобројна и реална аритметика
- Позициони запис бројева са датим бројем цифара и мешовите јединице мере
- Гранање 1 (гранања са сложеним условима и надовезана гранања)
- Основни алгоритми над малим серијама бројева
- Основни итеративни алгоритми 1 (са неугнежђеним петљама без гранања)
- Рад са текстуалним подацима (нискама)

Проблем. На једној далекој планети, једна недеља траје n дана и дани су нумерисани бројевима $1, 2, 3, \ldots, n$. Дане бројимо од првог дана прве недеље. Познато је да је данас дан са укупним редним бројем a.
Напиши програм који одређује који ће дан у недељи по реду бити за x дана (та недеља не мора бити прва недеља).
На улазу су редом бројеви n, a и x.

Улаз.

```
15 2 18
```

Излаз.

```
5
```

Објашњење. Данас је 2. дан, а за 18 дана ће бити 20. дан. Дани у другој недељи су 16, 17, 18, 19, 20, 21, ..., 29. и 30. Према томе, то ће бити пети дан те недеље.

Приметимо, дани прве недеље су нумерисани бројевима од 1 до n, дани друге недеље бројевима од $n+1$ до $2n$, дани треће недеље бројевима од $2n+1$ до $3n$ итд.

Приметимо, дани прве недеље су нумерисани бројевима од 1 до n, дани друге недеље бројевима од $n+1$ до $2n$, дани треће недеље бројевима од $2n+1$ до $3n$ итд.
Једно решење је да израчунамо којој недељи припада дан са редним бројем $a+x$, па да на основу тога одредимо који је то дан у тој недељи.

Приметимо, дани прве недеље су нумерисани бројевима од 1 до n, дани друге недеље бројевима од $n+1$ до $2n$, дани треће недеље бројевима од $2n+1$ до $3n$ итд.
Једно решење је да израчунамо којој недељи припада дан са редним бројем $a+x$, па да на основу тога одредимо који је то дан у тој недељи.
Други начин је да приметимо да је ово управо остатак при дељењу броја $a+x$ са n. Једини случај када треба бити пажљив је када је $a+x$ дељив са n. Тада је остатак при дељењу једнак 0, али је одговор n.

Целобројна и реална аритметика

Приметимо, дани прве недеље су нумерисани бројевима од 1 до n, дани друге недеље бројевима од $n+1$ до $2n$, дани треће недеље бројевима од $2n+1$ до $3n$ итд.
Једно решење је да израчунамо којој недељи припада дан са редним бројем $a+x$, па да на основу тога одредимо који је то дан у тој недељи.
Други начин је да приметимо да је ово управо остатак при дељењу броја $a+x$ са n. Једини случај када треба бити пажљив је када је $a+x$ дељив са n. Тада је остатак при дељењу једнак 0, али је одговор n.

```
int redniBrojDana = (a + x) % n;
if(redniBrojDana == 0)
    cout << n;
else
    cout << redniBrojDana;
```

Позициони запис бројева - конверзије

Проблем. Са стандардног улаза се учитава величина угла у степенима, минутима и секундама (угао је мањи од 180 степени, али је задат тако да број минута и број секунди може бити и већи од 59). Написати програм који одређује да ли је угао оштар, прав или туп.

Улаз.

```
89 79 3
```

Излаз.

```
tup
```

Позициони запис бројева - угао

```
int stepeni, minuti, sekundi;
cin >> stepeni >> minuti >> sekundi;
int ugaoNaUlazu = 3600 * stepeni + 60 * minuti + sekundi;
int pravUgao = 3600 * 90 + 60 * 0 + 0;
if(ugaoNaUlazu < pravUgao)
    cout << "ostar";
else if(ugaoNaUlazu == pravUgao)
    cout << "prav";
else
    cout << "tup";
```

На сличан начин се конвертује време (сати, минути, секунде у секунде).

Позициони запис бројева

Проблем. Наставник је поставио услов да на секцију могу да иду само они ученици који имају 5 из информатике и бар 4 из математике. У првом реду улаза је број N, број ученика. У сваком од следећих N редова низ оцена једног ученика из свих 11 предмета редом, без размака. Оцена из математике је шеста, а из информатике девета у низу. Колико ученика може да се пријави за секцију.

Улаз.

```
4
55555355455
55555455455
55555455555
22222522522
```

Излаз.

```
2
```

Позициони запис бројева - решење

Најједноставније је да учитавамо сваки ред као стринг од 11 карактера (цифара). Потребно је да пребројимо оне стрингове код којих је шеста цифра слева (бројећи од 1) већа од 3, а девета слева једнака 5.

```
string s; int brojac = 0;
for (int i = 0; i < n; i++){
    cin >> s;
    if (s[5] - '0' > 3 && s[8] - '0' == 5)
        br++;
}
```

```
brojac = 0
for i in range(n):
    s = input()
    if int(s[5]) > 3 and int(s[8]) == 5:
        br += 1
```

Филтрирање мале серије бројева

Проблем. За потребе овог задатка *трипл–дабл* дефинишемо као најмање двоцифрен учинак (10 или више) у бар три од пет категорија које се прате. Другим речима: бар три броја већа од 9 у једном реду улаза.
Написати програм који одређује колико пута је кошаркаш постигао такозвани трипл–дабл.

Улаз.

```
3
20 9 7 2 1
127 12 11 3 0
0 10 11 14 12
```

Излаз.

```
2
```

Филтрирање мале серије бројева

Да бисмо установили да ли неки ред улаза треба бројати, треба у сваком реду пребројати елементе који су већи од 9. То значи да ћемо у овом задатку применити технику пребројавања „на два нивоа”: глобално бројимо редове који испуњавају услов, а у сваком реду бројимо елементе веће од 9.

Филтрирање мале серије бројева

Да бисмо установили да ли неки ред улаза треба бројати, треба у сваком реду пребројати елементе који су већи од 9. То значи да ћемо у овом задатку применити технику пребројавања „на два нивоа”: глобално бројимо редове који испуњавају услов, а у сваком реду бројимо елементе веће од 9.

Један од начина да пребројимо елементе веће од 9 у сваком реду је да након уноса 5 бројева a, b, c, d, e ручно пребројимо колико их је већих од 9.

Филтрирање мале серије бројева

```
brTriplDabl = 0;
for (int utak = 0; utak < n; utak++){
    int brDvoc = 0;
    if(a > 9)
        brDvoc++;
    if(b > 9)
        brDvoc++;
    if(c > 9)
        brDvoc++;
    if(d > 9)
        brDvoc++;
    if(e > 9)
        brDvoc++;

    if(brDvoc >= 3)
        brTriplDabl++;
}
cout << brTriplDabl;
```

Филтрирање мале серије бројева - уопштење

```
int n, brTriplDabl = 0;
int a[5];
cin >> n;
for (int utak = 0; utak < n; utak++)
{
    cin >> a[0] >> a[1] >> a[2] >> a[3] >> a[4];
    int brDvoc = 0;
    for (int kateg = 0; kateg < 5; kateg++)
        if (a[kateg] > 9)
            brDvoc++;

    if (brDvoc >= 3)
        brTriplDabl++;
}
cout << brTriplDabl << endl;
```

Сложеност

- Ако за сортирање низа од n елемената треба t секунди, колико треба времена да се сортира низ од $2n$ елемената?

Сложеност

- Ако за сортирање низа од n елемената треба t секунди, колико треба времена да се сортира низ од $2n$ елемената?
- У C++ се може извршити око 10^8 операција у једној секунди.
  1. Битовске операције: and, or, xor, «, » $(2 \cdot 10^8)$
  2. +, -, * $(2 \cdot 10^8)$
  3. Приступ низу $(2 \cdot 10^8)$
  4. /, mod (нешто мање)
  5. Приступ хеш мапи, читање улаза, исписивање излаза (10^6)

```
int suma = 0; // 1 dodela
for (int i = 0; i < n; ++i) { // 1 dodela, n+1 provera
    suma = suma + i; // 1 sabiranje, 1 dodela
    // 1 inkrementiranje ++i
}
```

- $1 + 1 + (n + 1) + 3n = 4n + 3 \in \mathcal{O}(n)$ операција

```
int suma = 0; // 1 dodela
for (int i = 0; i < n; ++i) { // 1 dodela, n+1 provera
    suma = suma + i; // 1 sabiranje, 1 dodela
    // 1 inkrementiranje ++i
}
```

- $1 + 1 + (n + 1) + 3n = 4n + 3 \in \mathcal{O}(n)$ операција

```
int suma = 0; // 1 dodela
for (int i = 0; i < n; ++i) { // 1 dodela, n+1 provera
    for (int j = 0; j < n; ++j) { // 1 dodela, n+1 provera
        suma = suma + i; // 1 sabiranje, 1 dodela
        // 1 inkrementiranje ++j
    }
    // 1 inkrementiranje ++i
}
```

- $1 + 1 + (n + 1) + n \cdot (1 + (n + 1) + 3n) + n \in \mathcal{O}(n^2)$ операција

$$c \ll \log(n) \ll n^c \ll c^n \ll n!, c > 1$$

Сложеност	Назив	$t = 1s^*$
$\mathcal{O}(1)$	константна	/
$\mathcal{O}(\log(n))$	логаритамска	2^{10^8}
$\mathcal{O}(\sqrt{n})$	корена	10^{16}
$\mathcal{O}(n)$	линеарна, полиномијална	10^8
$\mathcal{O}(n \log n)$	логлинеарна	10^6
$\mathcal{O}(n^2)$	квадратна, полиномијална	10^4
$\mathcal{O}(n^3)$	кубна, полиномијална	464
$\mathcal{O}(n^k), k > 3$	полиномијална	$10^{\frac{8}{k}}$
$\mathcal{O}(c^n), c > 1$	експоненцијална	$\frac{18.4207}{\ln c}$
$\mathcal{O}(n!)$	факторијел	11

** Горња граница за n за коју се програм са $t(n) \in \mathcal{O}(t(n))$ инструкција изврши за највише 1s, под претпоставком да се у 1s изврши 10^8 инструкција.*

Проблем. Два госта су дошла у хотел и желе да одседну у собама које су што ближе једна другој. Ако постоји више таквих соба, они бирају да буду што даље од рецепције (собе са што већим редним бројем), како им бука не би сметала. Ако је познат списак слободних соба у том тренутку, одредити бројеве соба које гости треба да добију.

Улаз.

```
18 6 25 11 4 1 16
```

Излаз.

```
16 18
```

```
sort(begin(sobe), end(sobe));
int min = 1;
for (int i = 2; i < sobe.size(); ++i) {
    if (sobe[i] - sobe[i-1] <= sobe[min] - sobe[min-1])
        min = i;
}
int soba1 = sobe[min-1];
int soba2 = sobe[min];
```

- Сложеност: $\mathcal{O}(n \log n)$

Проблем. Дат је низ целих бројева. Одредити који број се најчешће појављује у низу.

Проблем. Дат је низ целих бројева. Одредити колико различитих елемената има у низу.

Проблем. Проверити да ли су дата два стринга анаграми.

Бинарна претрага елемента у низу

Проблем. Дат је сортиран низ пиродних бројева и природан број x. Одредити да ли се број x налази у низу.

Улаз.

```
2 9 11 18 20 45 60
45
```

Излаз.

```
Da
```

```
int l = 0, d = n - 1;
while (l <= d) {
    // nalazimo sredinu intervala
    int s = l + (d - l) / 2;
    if (x < a[s])
        d = s - 1;
    else if (x > a[s])
        l = s + 1;
    else
        // nasli smo element x na poziciji s
        return true;
}
// element ne postoji u nizu
return false;
```

- Сложеност: $\mathcal{O}(\log n)$

Проблем. Дат је цео број s и низ различитих целих бројева. Одредити број парова у низу који имају збир једнак броју s.

Проблем. Сортиран низ различитих целих бројева је ротиран k места удесно. Пронаћи најмањи елемент у том низу.

Проблем. Дате су две групе корисника А и Б који играју фудбалску игрицу. Сваки корисник има свој скор, природан број, који одређује колико је добар у игрици. Потребно је направити меч између два играча из супротних група тако да је разлика њихових скорова минимална.

Улаз.

```
2120 7940 11530 4680 17820
850 13420 5770 6300
```

Излаз.

```
4680
5770
```

```
sort(begin(a), end(a));
sort(begin(b), end(b));

int i = 0, j = 0;
int rezultat = numeric_limits<int>::max();

while (i < a.size() && j < b.size()) {
    if (a[i] <= b[j]) {
        rezultat = min(rezultat, b[j] - a[i]);
        ++i;
    } else {
        rezultat = min(rezultat, a[i] - b[j]);
        ++j;
    }
}
```

- Сложеност: $\mathcal{O}(n \log n + m \log m)$

Проблем. Дат је цео број s и низ различитих целих бројева. Одредити број парова у низу који имају збир једнак броју s.

Проблем. Дат је низ различитих целих бројева. Одредити на колико начина се могу изабрати три елемента из низа чији је збир једнак 0.

Проблем. У датом низу природних бројева пронаћи све сегменте чији је збир једнак датом броју.

Проблем. Дата су два стринга a и b. Одредити дужину најмањег сегмента стринга a у којем се појављују сви карактери из стринга b.

Покретни прозор фиксне ширине

Проблем. Група ученика је добила карте за позоришну представу. Све карте су за седишта у првом реду, али нису сва седишта једно поред другог. Ученици су одлучили да питају остале људе из реда да се замене са некима од њих како би они сви седели заједно. Одредити колико најмање замена треба да се направи ако је познат распоред седења у првом реду.

Улаз.

```
0 1 0 0 1 1 0 1 0 0
```

Излаз.

```
1
```

```
int ukupno_ucenika = ukupan_broj_jedinica(sedista);

// prozor sirine ukupno_ucenika
int trenutno_ucenika = 0;
for (int i = 0; i < ukupno_ucenika; ++i) {
    if (sedista[i] == 1)
        ++trenutno_ucenika;
}

int max_ucenika = trenutno_ucenika;

for (int i = 0; i < sedista.size() - ukupno_ucenika; ++i) {
    // pomeriti prozor desno za jedno mesto
    if (sedista[i] == 1)
        --trenutno_ucenika;
    if (sedista[i + ukupno_ucenika] == 1)
        ++trenutno_ucenika;
    if (trenutno_ucenika > max_ucenika)
        max_ucenika = trenutno_ucenika;
}

return ukupno_ucenika - max_ucenika;
```

- Сложеност: $\mathcal{O}(n)$

Проблем. Дат је низ реалних бројева и природан број k. Написати програм којим се у низу одређује сегмент дужине k са највећим просеком (ако више сегмената има исти просек, пријавити последњи од њих).

Проблем. Дата су два стринга a и b и природан број k. Одредити да ли у стрингу a постоји сегмент дужине k у којем се појављују сви карактери из стринга b.

Бинарна претрага по решењу

Проблем. За сваког од n играча је познато колико карата има у руци. На крају потреза свако ко има више од k карата мора да одбаци вишак тако да му остане тачно k карата. Укупан број одбачених карата свих играча треба да буде m. Одредити број k или исписати -1 ако решење не постоји.

Улаз.

```
24 21 19 14 22
14
```

Излаз.

```
18
```

```
int a = 0;
int b = najveci_element(karte);

while (a <= b) {
    int k = a + (b - a) / 2;
    int odbacene_karte = 0;
    for (int broj_karata : karte) {
        odbacene_karte += max(0, broj_karata - k);
    }
    if (odbacene_karte > m)
        a = k + 1;
    else if (odbacene_karte < m)
        b = k - 1;
    return k;
}

return -1;
```

- Сложеност: $\mathcal{O}(n \log M)$, M је највећи број карата у руци

Проблем. Дата су два стринга a и b. Одредити дужину најмањег сегмента стринга a у којем се појављују сви карактери из стринга b.

- Факторизација броја на чиниоце $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$;
- Број и збир делилаца природног броја;
- Израчунавање НЗД и НЗС два или више бројева (низа бројева);
- Еуклидов алгоритам;
- Ератостеново сито;

Факторизација броја

Проблем. Написати програм који за унети природни број n ($n \leqslant 10^{12}$) одређује најмањи природан број m такав да $n \cdot m$ има непаран број делилаца у скупу природних бројева (укључујући 1 и $n \cdot m$).

Улаз.

```
12
```

Излаз.

```
3
```

Објашњење. Делиоци броја $12 \cdot 3 = 36$ су $1, 2, 3, 4, 6, 9, 12, 18, 36$. Лако се види да бројеви $12 \cdot 1 = 12$ и $12 \cdot 2 = 24$ имају 6, односно 8 делиоца у скупу природних бројева.

Два наивна приступа

Два наивна приступа

Прва идеја. Испитати све бројеве $1, 2, 3, \ldots, n$ као кандидате за m, проверити петљом од 1 до $n \cdot m$ колико бројева у интервалу $[1, n \cdot m]$ дели број $n \cdot m$ и одредити најмањи m такав да непаран број бројева у интервалу $[1, n \cdot m]$ дели број $n \cdot m$. Ово решење ради у сложености $\mathcal{O}(n^3)$.

Два наивна приступа

Прва идеја. Испитати све бројеве $1, 2, 3, \ldots, n$ као кандидате за m, проверити петљом од 1 до $n \cdot m$ колико бројева у интервалу $[1, n \cdot m]$ дели број $n \cdot m$ и одредити најмањи m такав да непаран број бројева у интервалу $[1, n \cdot m]$ дели број $n \cdot m$. Ово решење ради у сложености $\mathcal{O}(n^3)$.

Друга идеја. Делиоце можемо избројати петљом од 1 до $\sqrt{n \cdot m}$. Ово решење ради у сложености $\mathcal{O}(n^2)$.

Кључна опсервација

Приметимо да важи: *да би број имао непаран број делилаца, он мора да буде квадрат природног броја* (и обрнуто).

Кључна опсервација

Приметимо да важи: *да би број имао непаран број делилаца, он мора да буде квадрат природног броја* (и обрнуто).

Први начин. Ако природан број d дели n, тада и $\frac{n}{d}$ дели n. Из тога можемо закључити да, уколико су за свако d, бројеви d и $\frac{n}{d}$ различити, сваки број d има свог пара, па би то резултирало парним бројем делилаца.

Бројеви су једнаки ако важи $n = d^2$, тј. ако је n квадрат природног броја (тада и даље сви делиоци осим $\sqrt{n}$ имају свог пара).

Кључна опсервација

Приметимо да важи: *да би број имао непаран број делилаца, он мора да буде квадрат природног броја* (и обрнуто).

Кључна опсервација

Приметимо да важи: *да би број имао непаран број делилаца, он мора да буде квадрат природног броја* (и обрнуто).
Други начин. Посматрајмо факторизацију броја $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$. Сваки делилац броја n има облик $d = p_1^{s_1} p_2^{s_2} \cdots p_k^{s_k}$, где је $0 \leqslant s_i \leqslant \alpha_i$ за свако $i = 1, 2, \ldots, k$. Сваки од експонената s_i може се изабрати на $\alpha_i + 1$ начина. То нам даје укупно $(\alpha_1 + 1)(\alpha_2 + 1) \cdots (\alpha_k + 1)$ могућности за делилац d. То је уједно и број делилаца броја n. Приметимо да је број делилаца непаран ако и само ако је $\alpha_i + 1$ непаран, односно α_i паран за свако $i = 1, 2, \ldots, k$. То управо значи да је n потпун квадрат.

Трећа идеја

Задатак смо свели на: *одредити најмање* m *такво да је* $n \cdot m$ *потпун квадрат*.

Трећа идеја

Задатак смо свели на: *одредити најмање* m *такво да је* $n \cdot m$ *потпун квадрат*.

Трећа идеја. Овај закључак намеће идеју да испитамо петљом све бројеве $1, 2, 3, \ldots, n$ као кандидате за m и проверимо да ли је тренутни број помножен са n потпун квадрат. Ово решење ради у сложености $\mathcal{O}(n)$.

Комплетно решење

Четврта идеја. Вратимо се на факторизацију броја $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$. Циљ нам је да број $n \cdot m$ буде потпун квадрат тј. да сви експоненти буду парни бројеви. Стога, број m ћемо потражити у облику $m = p_1^{\beta_1} p_2^{\beta_2} \cdots p_k^{\beta_k}$ (тада $\alpha_i + \beta_i$ треба да буде паран број за свако $i = 1, 2, \ldots, k$).

Комплетно решење

Четврта идеја. Вратимо се на факторизацију броја $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$. Циљ нам је да број $n \cdot m$ буде потпун квадрат тј. да сви експоненти буду парни бројеви. Стога, број m ћемо потражити у облику $m = p_1^{\beta_1} p_2^{\beta_2} \cdots p_k^{\beta_k}$ (тада $\alpha_i + \beta_i$ треба да буде паран број за свако $i = 1, 2, \ldots, k$).

Уколико је експонент (простог фактора p_i) α_i паран, тада ће m бити најмањи за $\beta_i = 0$, а уколико је експонент (простог фактора p_i) α_i непаран, тада ће m бити најмањи за $\beta_i = 1$.

Комплетно решење

Четврта идеја. Вратимо се на факторизацију броја $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$. Циљ нам је да број $n \cdot m$ буде потпун квадрат тј. да сви експоненти буду парни бројеви. Стога, број m ћемо потражити у облику $m = p_1^{\beta_1} p_2^{\beta_2} \cdots p_k^{\beta_k}$ (тада $\alpha_i + \beta_i$ треба да буде паран број за свако $i = 1, 2, \ldots, k$).

Уколико је експонент (простог фактора p_i) α_i паран, тада ће m бити најмањи за $\beta_i = 0$, а уколико је експонент (простог фактора p_i) α_i непаран, тада ће m бити најмањи за $\beta_i = 1$.

Дакле, m се добија као производ простих бројева са непарним експонентом у факторизацији броја n. Ово решење ради у сложености факторизације броја тј. $\mathcal{O}(\sqrt{n})$.

Имплементација

```
long long m = 1, p = 2;
while (p * p <= n) {
    // racunamo visestrukost cinioca p
    int k = 0;
    while (n % p == 0) {
        n /= p;
        k++;
    }
    // ako se cinilac p javlja neparan broj puta tada se
    ↪  on mora javiti u dopuni m
    if (k % 2 != 0)
        m *= p;
    p++;
}
// n se sveo na svoj najveci prost cinilac
if (n > 1)
    // i on mora da ucestvuje u dopuni (neparni eksponent)
    m *= n;
```

Разна својства природних бројева

Проблем. Лутрија се организује извлачењем N природних бројева, мањих од 10^{18}. Бројеви се могу понављати, а важан је и њихов редослед.

Пера је успео да дође до неких додатних информација о наредној добитној комбинацији. Нека су бројеви који ће бити извучени на лутрији $L_i, 1 \leqslant i \leqslant N$. Пера је сазнао низ који се састоји од $N-1$ броја, од којих i-ти број, A_i, представља највећи број који дели и L_i и L_{i+1}.

Сада Пера жели да попуни листић и за то му је потребна помоћ. Исписати једну комбинацију која испуњава услове, или -1, уколико таква комбинација не постоји. Уколико постоји више комбинација које испуњавају услове, исписати било коју.

Примери

Улаз.

```
4
3 4 10
```

Излаз.

```
3 12 20 10
```

Улаз.

```
4
3 4 6
```

Излаз.

```
-1
```

Зашто други пример нема решење?

Улаз.

```
4
3 4 6
```

Приметимо да важи

1. $3 \mid L_1$ и $3 \mid L_2$;

Зашто други пример нема решење?

Улаз.

```
4
3 4 6
```

Приметимо да важи

1. $3 \mid L_1$ и $3 \mid L_2$;
2. $4 \mid L_2$ и $4 \mid L_3$;

Зашто други пример нема решење?

Улаз.

```
4
3 4 6
```

Приметимо да важи

1. $3 \mid L_1$ и $3 \mid L_2$;
2. $4 \mid L_2$ и $4 \mid L_3$;
3. $6 \mid L_3$ и $6 \mid L_4$;

Зашто други пример нема решење?

Улаз.

```
4
3 4 6
```

Приметимо да важи

1. $3 \mid L_1$ и $3 \mid L_2$;
2. $4 \mid L_2$ и $4 \mid L_3$;
3. $6 \mid L_3$ и $6 \mid L_4$;

Из 1. и 2. добијамо $12 \mid L_2$, а из 2. и 3. имамо $12 \mid L_3$.

Зашто други пример нема решење?

Улаз.

```
4
3 4 6
```

Приметимо да важи

1. $3 \mid L_1$ и $3 \mid L_2$;
2. $4 \mid L_2$ и $4 \mid L_3$;
3. $6 \mid L_3$ и $6 \mid L_4$;

Из 1. и 2. добијамо $12 \mid L_2$, а из 2. и 3. имамо $12 \mid L_3$.
Међутим, на улазу је дато да је 4 највећи број који дели L_2 и L_3.
Ово разматрање можемо да уопштимо.

Уопштење

Приметимо да важи:

Уопштење

Приметимо да важи:
$A_1 \mid L_1$ и $A_1 \mid L_2$;

Уопштење

Приметимо да важи:
$A_1 \mid L_1$ и $A_1 \mid L_2$;
$A_2 \mid L_2$ и $A_2 \mid L_3$;

Уопштење

Приметимо да важи:
$A_1 \mid L_1$ и $A_1 \mid L_2$;
$A_2 \mid L_2$ и $A_2 \mid L_3$;
$A_3 \mid L_3$ и $A_3 \mid L_4$;

Уопштење

Приметимо да важи:

$A_1 \mid L_1$ и $A_1 \mid L_2$;

$A_2 \mid L_2$ и $A_2 \mid L_3$;

$A_3 \mid L_3$ и $A_3 \mid L_4$;

$\vdots$

$A_{n-1} \mid L_{n-1}$ и $A_{n-1} \mid L_n$;

Уопштење

Приметимо да важи:
$A_1 \mid L_1$ и $A_1 \mid L_2$;
$A_2 \mid L_2$ и $A_2 \mid L_3$;
$A_3 \mid L_3$ и $A_3 \mid L_4$;
$\vdots$
$A_{n-1} \mid L_{n-1}$ и $A_{n-1} \mid L_n$;

Јасно, одавде добијамо

$$A_1 \mid L_1, \text{НЗС}(A_1, A_2) \mid L_2, \text{НЗС}(A_2, A_3) \mid L_3, \ldots,$$

$$\text{НЗС}(A_{n-2}, A_{n-1}) \mid L_{n-1} \text{ и } A_{n-1} \mid L_n.$$

Уопштење

Приметимо да важи:
$A_1 \mid L_1$ и $A_1 \mid L_2$;
$A_2 \mid L_2$ и $A_2 \mid L_3$;
$A_3 \mid L_3$ и $A_3 \mid L_4$;
$\vdots$
$A_{n-1} \mid L_{n-1}$ и $A_{n-1} \mid L_n$;

Јасно, одавде добијамо

$$A_1 \mid L_1, \text{НЗС}(A_1, A_2) \mid L_2, \text{НЗС}(A_2, A_3) \mid L_3, \ldots,$$

$$\text{НЗС}(A_{n-2}, A_{n-1}) \mid L_{n-1} \text{ и } A_{n-1} \mid L_n.$$

Нека је $B_1 = A_1, B_2 = \text{НЗС}(A_1, A_2), B_3 = \text{НЗС}(A_2, A_3), \ldots,$
$B_{n-1} = \text{НЗС}(A_{n-2}, A_{n-1})$ и $B_n = A_{n-1}$.

Решење

$B_1 = A_1, B_2 = \mathsf{НЗС}(A_1, A_2), B_3 = \mathsf{НЗС}(A_2, A_3), \ldots,$
$B_{n-1} = \mathsf{НЗС}(A_{n-2}, A_{n-1})$ и $B_n = A_{n-1}$.

Решење

$B_1 = A_1, B_2 = \text{НЗС}(A_1, A_2), B_3 = \text{НЗС}(A_2, A_3), \ldots,$
$B_{n-1} = \text{НЗС}(A_{n-2}, A_{n-1})$ и $B_n = A_{n-1}$.

Прво, потребно је проверити да ли ови НЗС одговарају бројевима из поставке. Уколико добијемо да је неки НЗС већи од броја из поставке, онда задатак нема решења.

Решење

$B_1 = A_1, B_2 = \text{НЗС}(A_1, A_2), B_3 = \text{НЗС}(A_2, A_3), \ldots,$
$B_{n-1} = \text{НЗС}(A_{n-2}, A_{n-1})$ и $B_n = A_{n-1}$.

Прво, потребно је проверити да ли ови НЗС одговарају бројевима из поставке. Уколико добијемо да је неки НЗС већи од броја из поставке, онда задатак нема решења.

Уколико се сви НЗС уклапају у бројеве из поставке, довољно је да приметимо да низ B по конструкцији задовољава услове задатка и можемо њега да одштампамо на излаз.

Ефикасно рачунање НЗС

За два природна броја a и b је познато да важи

$$\text{НЗС}(a, b) \cdot \text{НЗД}(a, b) = a \cdot b.$$

Ефикасно рачунање НЗС

За два природна броја a и b је познато да важи

$$\text{НЗС}(a, b) \cdot \text{НЗД}(a, b) = a \cdot b.$$

НЗД два броја можемо израчунати на више начина. Најпознатији алгоритам је Еуклидов:

$$\text{НЗД}(a, b) = \begin{cases} a & b = 0; \\ \text{НЗД}(b, b\%a) & b \neq 0. \end{cases}$$

```
int nzd(int a, int b){
    if(b == 0)
        return a;
    return nzd(b, a%b);
}
```

Рачунање НЗС

За два природна броја a и b је познато да важи

$$\text{НЗС}(a, b) \cdot \text{НЗД}(a, b) = a \cdot b.$$

C++ има уграђену функцију у библиотеци algorithm

```
    nzd = __gcd(a, b);
```

Python има уграђену функцију у библиотеци math

```
math.gcd(int1, int2)
```

Greedy (грамзиви) алгоритми

Алгоритми код којих се у сваком кораку узима *локално оптимално* решење и који гарантују да ће такви избори на крају довести до *глобално оптималног* решења називају се похлепни или грамзиви алгоритми (енгл. greedy algorithms).

Прорeђен скуп

Проблем. Дат је скуп целих бројева дужине n. Написати програм који одређује величину највећег подскупа таквог да не садржи два елемента чија је разлика строго мања d.

Улаз.

```
7 3
8 2 4 1 9 6 5
```

Излаз.

```
3
```

Објашњење. Постоји више подскупова величине 3 који испуњавају тражени услов, а не постоји такав подскуп величине 4. Један такав подскуп је $\{8, 2, 5\}$, зато што се свака два његова елемента разликују за најмање 3.

Решење

Задатак решавамо тако што низ прво сортирамо, а затим проређен скуп иницијализујемо на први елемент низа и у сваком кораку га проширујемо најмањим елементом низа који је на растојању већем или једнаком од d у односу на највећи елемент до тада изграђеног скупа.

Решење

Задатак решавамо тако што низ прво сортирамо, а затим проређен скуп иницијализујемо на први елемент низа и у сваком кораку га проширујемо најмањим елементом низа који је на растојању већем или једнаком од d у односу на највећи елемент до тада изграђеног скупа.

Сортирани низ из примера је $1, 2, 4, 5, 6, 8, 9$. Скуп иницијализујемо на $\{1\}$, а затим га проширујемо редом елементима 4 и 8, добијајући најдужи могући проређени скуп $\{1, 4, 8\}$.

Решење

Задатак решавамо тако што низ прво сортирамо, а затим прорeђен скуп иницијализујемо на први елемент низа и у сваком кораку га проширујемо најмањим елементом низа који је на растојању већем или једнаком од d у односу на највећи елемент до тада изграђеног скупа.

Сортирани низ из примера је $1, 2, 4, 5, 6, 8, 9$. Скуп иницијализујемо на $\{1\}$, а затим га проширујемо редом елементима 4 и 8, добијајући најдужи могући прорeђени скуп $\{1, 4, 8\}$.

```
int prethodni = 0, velicina  = 1;
for(int i = 1; i < n; i++) {
    if (a[i] - a[prethodni] >= d) {
        prethodni = i;
        velicina++;
    }
}
```

Шта када похлепна стратегија закаже?

Проблем. Нека је дат скуп вредности апоена и циљна сума новца n. Треба да конструишемо збир n коришћењем најмањег могућег броја новчаница.

Шта када похлепна стратегија закаже?

Проблем. Нека је дат скуп вредности апоена и циљна сума новца n. Треба да конструишемо збир n коришћењем најмањег могућег броја новчаница.

Апоени су $\{1, 2, 5\}$ и $n = 12$, оптимално је $5 + 5 + 2 = 12$.

Шта када похлепна стратегија закаже?

Проблем. Нека је дат скуп вредности апоена и циљна сума новца n. Треба да конструишемо збир n коришћењем најмањег могућег броја новчаница.

Апоени су $\{1, 2, 5\}$ и $n = 12$, оптимално је $5 + 5 + 2 = 12$.
Ово намеће интуитиван начин за решавање: Увек изабери новчић највеће могуће вредности, али тако да збир вредности новчића не прелази тражени збир.

Шта када похлепна стратегија закаже?

Проблем. Нека је дат скуп вредности апоена и циљна сума новца n. Треба да конструишемо збир n коришћењем најмањег могућег броја новчаница.

Апоени су $\{1, 2, 5\}$ и $n = 12$, оптимално је $5 + 5 + 2 = 12$.
Ово намеће интуитиван начин за решавање: Увек изабери новчић највеће могуће вредности, али тако да збир вредности новчића не прелази тражени збир.

Међутим, испоставља се да овакав приступ **не** функционише увек. На пример, уколико је скуп апоена $\{1, 3, 4\}$ и $n = 6$, похлепни алгоритам би вратио $4 + 1 + 1 = 6$. Но, у овом случају оптимално је $3 + 3 = 6$.

Шта када похлепна стратегија закаже?

Проблем. Нека је дат скуп вредности апоена и циљна сума новца n. Треба да конструишемо збир n коришћењем најмањег могућег броја новчаница.

Апоени су $\{1, 2, 5\}$ и $n = 12$, оптимално је $5 + 5 + 2 = 12$.
Ово намеће интуитиван начин за решавање: Увек изабери новчић највеће могуће вредности, али тако да збир вредности новчића не прелази тражени збир.

Међутим, испоставља се да овакав приступ **не** функционише увек. На пример, уколико је скуп апоена $\{1, 3, 4\}$ и $n = 6$, похлепни алгоритам би вратио $4 + 1 + 1 = 6$. Но, у овом случају оптимално је $3 + 3 = 6$.

Овај задатак је познат под именом *change-making problem* и у општем случају се решава методом динамичког програмирања.

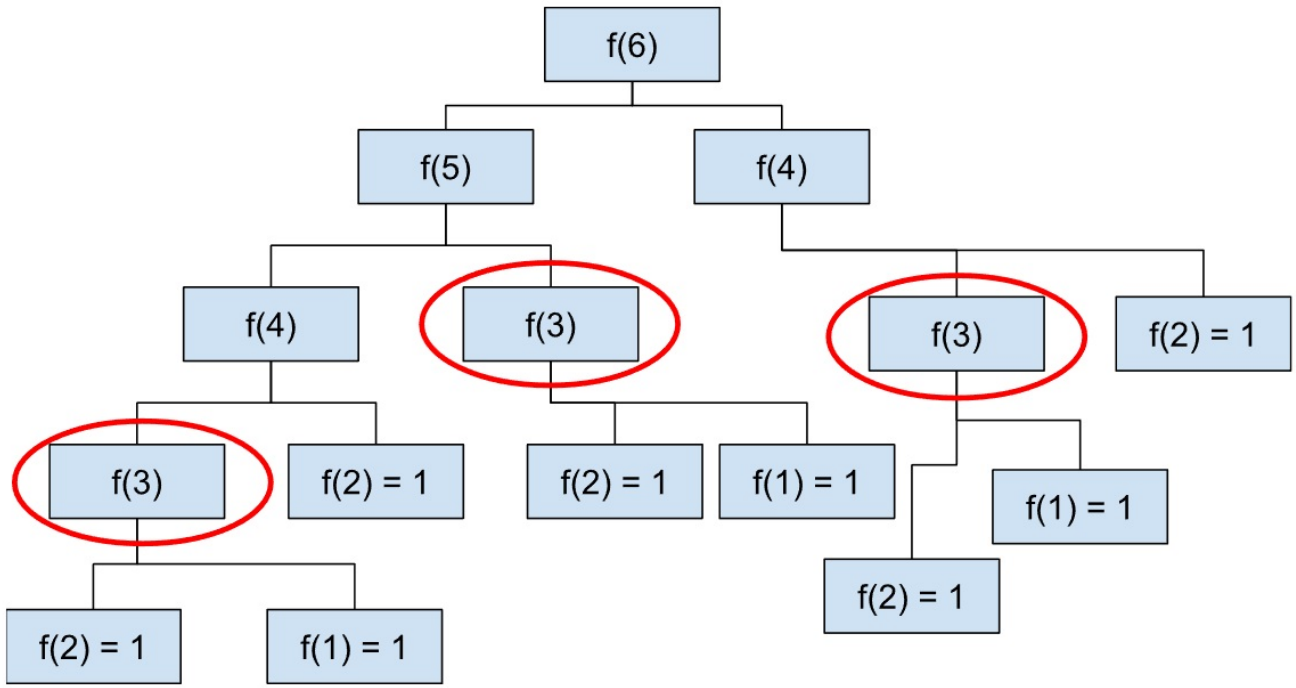
Динамичко програмирање

- Оптимизација рекурзивних функција чувањем међурезултата
- Ипробавање свих могућности, али на паметан начин
- Разлика у односу на „завади па владај” приступ: Потпроблеми се преклапају
- Приступ може бити одоздо на горе или одого на горе (мемоизација)
- Како да препознам да се проблем може решити динамичким програмирањем?
- Који је добар уводни проблем за динамичко програмирање?

Проблем Фибоначијевог низа

Нека су прва два члана неког низа 1 и 1. Сваки наредни члан се дефинише као збир претходна два.

- Овај проблем се често користи као уводни проблем за рекурзију, али њега је много природније решавати динамички.



Задатак: Сечење штапа

Дат је штап дужине n метара. Потребно је пресећи га на неколико места тако да добијени делови буду целобројне дужине и ниједан од добијених делова не буде дужи од k метара. Написати програм који одређује на колико начина је могуће извршити овакво сечење.

Улаз: Са стандардног улаза се уносе бројеви n $(1 \leq n \leq 1000)$ и k $(1 \leq k \leq 300)$.

Излаз: На стандардни излаз исписати број могућих сечења. Како тај број може бити велик, исписати његов остатак при дељењу са $10^9 + 9$.

Пример:

Улаз: 4 2

Излаз: 5

Објашњење: Могућа су следећа сечења представљена дужинама добијених делова штапа: 1 1 1 1; 2 1 1; 1 2 1; 1 1 2; 2 2

Сечење штапа - решење

```cpp
#include <iostream>
#include <vector>
using namespace std;
const int mod = 1000000009;
int main()
{
    int n, k;
    cin >> n >> k;
    vector<int> broj(n + 1);
    broj[0] = 1;
    int zbirPoslednjihK = 1;
    for (int i = 1; i <= n; i++) {
        broj[i] = zbirPoslednjihK;
        zbirPoslednjihK = (zbirPoslednjihK + broj[i]) % mod;
        if (i >= k)
            zbirPoslednjihK = (zbirPoslednjihK + mod - broj[i
            ↪  - k]) % mod;
        }
    cout << broj[n] << '\n';
    return 0;
```