LOBACHEVSKY
UNIVERSITY

Modern C++

2. Task: Sqrt

Compiling and linking (30.10.2017)

Sidnev A.A.

Sqrt(x). Version 1

LOBACHEVSKY UNIVERSITY

```
class Solution {
  vector<int> sqr;
public:
  Solution() : sqr(46341) {
    iota(sqr.begin(), sqr.end(), 0);
  }
  int mySqrt(int x) {
    if (x < 0) {
      return INT_MIN;
    }
    auto root = upper_bound(sqr.begin(), sqr.end(), x,
                            [](const int& x, const int& i) {
      return x < i * i;
    });
    return *(--root);
  }
};
```

Sqrt(x). Version 2

LOBACHEVSKY UNIVERSITY

```
class Solution {
  vector<int> init() {
    vector<int> sqr(46341);
    iota(sqr.begin(), sqr.end(), 0);
    return sqr;
  }

public:
  int mySqrt(int x) {
    static vector<int> sqr = init();

    if (x < 0) {
      return INT_MIN;
    }

    auto root = upper_bound(sqr.begin(), sqr.end(), x, [](const int& x, const int& i) {
      return x < i * i;
    });

    return *(--root);
  }
};
```

Sqrt(x). Version 3

LOBACHEVSKY UNIVERSITY

```
class Solution {
  array<int, 46341> init() {
    array<int, 46341> sqr;
    iota(sqr.begin(), sqr.end(), 0);
    return sqr;
  }

public:
  int mySqrt(int x) {
    static array<int, 46341> sqr = init();

    if (x < 0) {
      return INT_MIN;
    }

    auto root = upper_bound(sqr.begin(), sqr.end(), x, [](const int& x, const int& i) {
      return x < i * i;
    });

    return *(--root);
  }
};
```

Sqrt(x). Version 4

```
class Solution {
public:
  int mySqrt(int x) {
    if (x < 0) {
      return INT_MIN;
    }
    else {
      int l = 0;
      int r = 46340;

      while (l < r) {
        int m = l + (r - l) / 2,
            p = m * m;
        if (p == x) {
          return m;
        } else if (p < x) {
          l = m + 1;
        } else {
          r = m;
        }
      }
      if (l * l > x) {
            return l - 1;
          } else {
            return l;
          }
      }
    }
};
```

Compilation and linking

LOBACHEVSKY UNIVERSITY

```cpp
// A.cpp -> A.obj
#include <iostream>
using std::cout;
using std::endl;

int x1;
int x2 = 1;
extern int x3;
extern int x4;
extern const int x5;

int f() {
  cout << x1 << endl;  // ???
  cout << x2 << endl;  // ???
  cout << x3 << endl;  // ???
  cout << x4 << endl;  // ???
  cout << x5 << endl;  // ???
  return 5;
}
```

```cpp
// B.cpp -> B.obj
float x1;
int x3 = 2;
int x4;
extern const int x5 = 1;

int f();

int main() {
  x1 = f();
}
```

Namespace

```
// A.cpp -> A.obj
#include <iostream>
using std::cout;
using std::endl;

namespace A {
int x1;
int x2 = 1;
extern int x3;
extern int x4;
extern const int x5;

int f() {
  cout << x1 << endl;  // ???
  cout << x2 << endl;  // ???
  cout << x3 << endl;  // ???
  cout << x4 << endl;  // ???
  cout << x5 << endl;  // ???
  return 5;
}
}  // namespace A
```

const is visible outside module with **extern**

```
// B.cpp -> B.obj
float x1;

namespace A {
int x3 = 2;
int x4;
extern const int x5 = 1;

int f();
}  // namespace A

int main() {
  x1 = A::f();
}
```

Static

```
// A.cpp -> A.obj
#include <iostream>
using std::cout;
using std::endl;

static int x1;
int x2 = 1;
extern int x3;
extern int x4;
extern const int x5;

int f() {
  cout << x1 << endl;  // 0
  cout << x2 << endl;  // 1
  cout << x3 << endl;  // 2
  cout << x4 << endl;  // 0
  cout << x5 << endl;  // 1
  return 5;
}
```

```
// B.cpp -> B.obj
float x1;
int x3 = 2;
int x4;
extern const int x5 = 1;

int f();

int main() {
  x1 = f();
}
```