高精度加法：

Yxc：

不圧位代码

#include <iostream>

#include <vector>

using namespace std;

vector<int> add(vector<int> &A, vector<int> &B)

{

if (A.size() < B.size()) return add(B, A);

vector<int> C;

int t = 0;

for (int i = 0; i < A.size(); i ++ )

{

t += A[i];

if (i < B.size()) t += B[i];

C.push\_back(t % 10);

t /= 10;

}

if (t) C.push\_back(t);

return C;

}

int main()

{

string a, b;

vector<int> A, B;

cin >> a >> b;

for (int i = a.size() - 1; i >= 0; i -- ) A.push\_back(a[i] - '0');

for (int i = b.size() - 1; i >= 0; i -- ) B.push\_back(b[i] - '0');

auto C = add(A, B);

for (int i = C.size() - 1; i >= 0; i -- ) cout << C[i];

cout << endl;

return 0;

}

压9位的代码

#include <iostream>

#include <vector>

using namespace std;

const int base = 1000000000;

vector<int> add(vector<int> &A, vector<int> &B)

{

if (A.size() < B.size()) return add(B, A);

vector<int> C;

int t = 0;

for (int i = 0; i < A.size(); i ++ )

{

t += A[i];

if (i < B.size()) t += B[i];

C.push\_back(t % base);

t /= base;

}

if (t) C.push\_back(t);

return C;

}

int main()

{

string a, b;

vector<int> A, B;

cin >> a >> b;

for (int i = a.size() - 1, s = 0, j = 0, t = 1; i >= 0; i -- )

{

s += (a[i] - '0') \* t;

j ++, t \*= 10;

if (j == 9 || i == 0)

{

A.push\_back(s);

s = j = 0;

t = 1;

}

}

for (int i = b.size() - 1, s = 0, j = 0, t = 1; i >= 0; i -- )

{

s += (b[i] - '0') \* t;

j ++, t \*= 10;

if (j == 9 || i == 0)

{

B.push\_back(s);

s = j = 0;

t = 1;

}

}

auto C = add(A, B);

cout << C.back();

for (int i = C.size() - 2; i >= 0; i -- ) printf("%09d", C[i]);

cout << endl;

return 0;

}

作者：yxc

链接：https://www.acwing.com/activity/content/code/content/39792/

来源：AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

蒟蒻：

#include <iostream>

#include <vector>

using namespace std;

string a, b;

// add

vector<int> add(vector<int> &A, vector<int> &B)

{

if (A.size() < B.size())

return add(B, A);

vector<int> c;

int t = 0;

for (int i = 0; i < A.size(); i++)

{

t += A[i];

if (i < B.size())

t += B[i];

c.push\_back(t % 10);

t /= 10;

}

if (t)

c.push\_back(t);

return c;

}

int main()

{

cin >> a >> b;

vector<int> A, B;

for (int i = a.size() - 1; i >= 0; i--)

A.push\_back(a[i] - '0');

for (int i = b.size() - 1; i >= 0; i--)

B.push\_back(b[i] - '0');

auto c = add(A, B);

for (int i = c.size() - 1; i >= 0; i--)

cout << c[i];

cout << endl;

return 0;

}

String版：

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

// C = A + B;

string add(const string& A, const string& B)

{

string C;

int t = 0;

for (int i = A.size()-1, j = B.size()-1; i >= 0 || j >= 0 || t > 0; i--, j--)

{

if (i >= 0) t += (A[i] - '0');

if (j >= 0) t += (B[j] - '0');

C += ((t % 10) + '0');

t /= 10;

}

reverse(C.begin(), C.end());

return C;

}

int main()

{

ios::sync\_with\_stdio(false);

string A, B;

cin >> A >> B;

cout << add(A, B) << endl;

return 0;

}

作者：二月

链接：https://www.acwing.com/solution/content/5692/

来源：AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

高精度减法：

思路

和高精度加法差不多，值得注意的是

减法的借位处理

相减为负数的处理

前导0的处理

收获

对于 t = A[i] - B[i] - t； 可以拆为 t = A[i] - t如果B[i]合法，再t -= B[i] 这么两步来做

相减后t的处理 ，把 t >=0 和 t < 0 用一个式子来表示 t = (t + 10) % 10 这个木有想到

A B大小判断，自己写的太冗余，不如单独拎出来

bool cmp(vector<int>& A, vector<int> &B)

{

if(A.size() != B.size()) return A.size() > B.size(); //直接ruturn 了就不用else

for(int i = A.size(); i >= 0; i--)

if(A[i] != B[i])

return A[i] > B[i];

return true;

}

拓展

A B 可以为正 为负怎么处理

绝对值的相减相加，再加标志，标志由输入来决定，判断一下字符串的首字符了类型

YXC

// 给定两个正整数

#include <iostream>

#include <vector>

using namespace std;

bool cmp(vector<int>& A, vector<int> &B)

{

if(A.size() != B.size()) return A.size() > B.size(); //直接ruturn 了就不用else

for(int i = A.size(); i >= 0; i--)

if(A[i] != B[i])

return A[i] > B[i];

return true;

}

vector <int> sub(vector<int>& A, vector<int> &B)

{

vector<int> C;

int t = 0;

for(int i = 0; i < A.size(); i++)

{

t = A[i] - t;

if(i < B.size()) t -= B[i];

C.push\_back((t + 10) % 10 ); // 合而为1

if(t < 0) t = 1;

else t = 0;

}

while(C.size() > 1 && C.back() == 0) C.pop\_back(); //去掉前导0

return C;

}

int main()

{

string a ,b;

vector<int> A, B;

cin >> a >> b ;

for(int i = a.size() - 1; i >= 0; i--) A.push\_back(a[i] - '0');

for(int i = b.size() - 1; i >= 0; i--) B.push\_back(b[i] - '0');

if (cmp(A,B))

{

auto C = sub(A, B);

for(int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);

return 0;

}

else

{

auto C = sub(B, A);

printf("-");

for(int i = C.size() - 1; i >= 0; i--) printf("%d", C[i]);

return 0;

}

}

作者：jasonlin

链接：https://www.acwing.com/solution/content/10973/

来源：AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

蒟蒻：

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

string sub(const string& A, const string& B)

{

string C;

int t = 0;

for (int i = A.size()-1, j = B.size()-1; i >= 0 || j >= 0 || t > 0; i--, j--)

{

if (i >= 0) t = (A[i] - '0') - t;

if (j >= 0) t -= (B[j] - '0');

C += ((t + 10) % 10 + '0');

if (t < 0) t = 1;

else t = 0;

}

while (C.size() > 1 && C.back() == '0') C.pop\_back();

reverse(C.begin(), C.end());

return C;

}

bool cmp(const string& A, const string& B)

{

if (A.size() != B.size()) return A.size() > B.size();

for (int i = 0; i < A.size(); i++)

if (A[i] != B[i]) return A[i] > B[i];

return true;

}

int main ()

{

string A, B;

cin >> A >> B;

if (cmp(A, B)) cout << sub(A, B) << endl;

else cout << "-" << sub(B, A) << endl;

return 0;

}

高乘

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

// res = A \* B

string mul(const string& A, const int b)

{

string res;

for (int i = A.size() - 1, t = 0; i >= 0 || t > 0; i--)

{

if (i >= 0) t += (A[i] - '0') \* b;

res += (t % 10) + '0';

t /= 10;

}

reverse(res.begin(), res.end());

return res;

}

int main ()

{

ios::sync\_with\_stdio(false);

string A;

int b;

cin >> A >> b;

cout << mul(A, b) << endl;

return 0;

}

作者：二月

链接：https://www.acwing.com/solution/content/5717/

来源：AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

高除

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

// res = A / b, r = A % b;

string div(const string& A, const int& b, int& r)

{

string res;

for (int i = 0; i < A.size(); i++)

{

r = r \* 10 + (A[i] - '0');

res += (r / b) + '0';

r %= b;

}

while (res.size() > 1 && res.front() == '0') res = res.substr(1);

return res;

}

int main ()

{

ios::sync\_with\_stdio(false);

string A;

int b;

cin >> A >> b;

int r = 0;

cout << div(A, b, r) << endl;

cout << r << endl;

return 0;

}

作者：二月

链接：https://www.acwing.com/solution/content/5718/

来源：AcWing

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。