Disassemble main

push %rbp

mov %rsp,%rbp

sub $0x20,%rsp

mov %fs:0x28,%rax

mov %rax,-0x8(%rbp)

xor %eax,%eax

movl $0x0,-0x14(%rbp)

mov -0x14(%rbp),%eax

cmp $0x5,%eax

je 0x308 <main+655>

lea 0x0(%rip),%rsi # 0xaa <main+49>

lea 0x0(%rip),%rdi # 0xb1 <main+56>

callq 0xb6 <main+61>

lea 0x0(%rip),%rsi # 0xbd <main+68>

lea 0x0(%rip),%rdi # 0xc4 <main+75>

callq 0xc9 <main+80>

lea 0x0(%rip),%rsi # 0xd0 <main+87>

lea 0x0(%rip),%rdi # 0xd7 <main+94>

callq 0xdc <main+99>

lea 0x0(%rip),%rsi # 0xe3 <main+106>

lea 0x0(%rip),%rdi # 0xea <main+113>

callq 0xef <main+118>

lea 0x0(%rip),%rsi # 0xf6 <main+125>

lea 0x0(%rip),%rdi # 0xfd <main+132>

callq 0x102 <main+137>

lea 0x0(%rip),%rsi # 0x109 <main+144>

lea 0x0(%rip),%rdi # 0x110 <main+151>

callq 0x115 <main+156>

lea -0x14(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi # 0x123 <main+170>

callq 0x128 <main+175>

mov -0x14(%rbp),%eax

cmp $0x1,%eax

jne 0x1c9 <main+336>

lea -0x10(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi # 0x142 <main+201>

callq 0x147 <main+206>

mov %rax,%rdx

lea -0xc(%rbp),%rax

mov %rax,%rsi

mov %rdx,%rdi

callq 0x159 <main+224>

mov -0x10(%rbp),%eax

mov %eax,%esi

lea 0x0(%rip),%rdi # 0x165 <main+236>

callq 0x16a <main+241>

lea 0x0(%rip),%rsi # 0x171 <main+248>

mov %rax,%rdi

callq 0x179 <main+256>

mov %rax,%rdx

mov -0xc(%rbp),%eax

mov %eax,%esi

mov %rdx,%rdi

callq 0x189 <main+272>

lea 0x0(%rip),%rsi # 0x190 <main+279>

mov %rax,%rdi

callq 0x198 <main+287>

mov %rax,%rcx

mov -0x10(%rbp),%edx

mov -0xc(%rbp),%eax

sub %eax,%edx

mov %edx,%eax

mov %eax,%esi

mov %rcx,%rdi

callq 0x1af <main+310>

mov %rax,%rdx

mov 0x0(%rip),%rax # 0x1b9 <main+320>

mov %rax,%rsi

mov %rdx,%rdi

callq 0x1c4 <main+331>

jmpq 0x97 <main+30>

mov -0x14(%rbp),%eax

cmp $0x2,%eax

jne 0x268 <main+495>

lea -0x10(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi # 0x1e3 <main+362>

callq 0x1e8 <main+367>

mov %rax,%rdx

lea -0xc(%rbp),%rax

mov %rax,%rsi

mov %rdx,%rdi

callq 0x1fa <main+385>

mov -0x10(%rbp),%eax

mov %eax,%esi

lea 0x0(%rip),%rdi # 0x206 <main+397>

callq 0x20b <main+402>

lea 0x0(%rip),%rsi # 0x212 <main+409>

mov %rax,%rdi

callq 0x21a <main+417>

mov %rax,%rdx

mov -0xc(%rbp),%eax

mov %eax,%esi

mov %rdx,%rdi

callq 0x22a <main+433>

lea 0x0(%rip),%rsi # 0x231 <main+440>

mov %rax,%rdi

callq 0x239 <main+448>

mov %rax,%rcx

mov -0x10(%rbp),%edx

mov -0xc(%rbp),%eax

add %edx,%eax

mov %eax,%esi

mov %rcx,%rdi

callq 0x24e <main+469>

mov %rax,%rdx

mov 0x0(%rip),%rax # 0x258 <main+479>

mov %rax,%rsi

mov %rdx,%rdi

callq 0x263 <main+490>

jmpq 0x97 <main+30>

mov -0x14(%rbp),%eax

cmp $0x3,%eax

jne 0x97 <main+30>

lea -0x10(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi # 0x282 <main+521>

callq 0x287 <main+526>

mov %rax,%rdx

lea -0xc(%rbp),%rax

mov %rax,%rsi

mov %rdx,%rdi

callq 0x299 <main+544>

mov -0x10(%rbp),%eax

mov %eax,%esi

lea 0x0(%rip),%rdi # 0x2a5 <main+556>

callq 0x2aa <main+561>

lea 0x0(%rip),%rsi # 0x2b1 <main+568>

mov %rax,%rdi

callq 0x2b9 <main+576>

mov %rax,%rdx

mov -0xc(%rbp),%eax

mov %eax,%esi

mov %rdx,%rdi

callq 0x2c9 <main+592>

lea 0x0(%rip),%rsi # 0x2d0 <main+599>

mov %rax,%rdi

callq 0x2d8 <main+607>

mov %rax,%rcx

mov -0x10(%rbp),%eax

mov -0xc(%rbp),%esi

cltd

idiv %esi

mov %eax,%esi

mov %rcx,%rdi

callq 0x2ee <main+629>

mov %rax,%rdx

mov 0x0(%rip),%rax # 0x2f8 <main+639>

mov %rax,%rsi

mov %rdx,%rdi

callq 0x303 <main+650>

jmpq 0x97 <main+30>

mov $0x0,%eax

mov -0x8(%rbp),%rcx

xor %fs:0x28,%rcx

je 0x321 <main+680>

callq 0x321 <main+680>

leaveq

retq

Disassemble again with markers

0x0000000000000079 <+0>: push rbp

0x000000000000007a <+1>: mov rbp,rsp

0x000000000000007d <+4>: sub rsp,0x20

0x0000000000000081 <+8>: mov rax,QWORD PTR fs:0x28

0x000000000000008a <+17>: mov QWORD PTR [rbp-0x8],rax

0x000000000000008e <+21>: xor eax,eax

0x0000000000000090 <+23>: mov DWORD PTR [rbp-0x14],0x0

0x0000000000000097 <+30>: mov eax,DWORD PTR [rbp-0x14]

0x000000000000009a <+33>: cmp eax,0x5

0x000000000000009d <+36>: je 0x308 <main+655>

0x00000000000000a3 <+42>: lea rsi,[rip+0x0] # 0xaa <main+49>

0x00000000000000aa <+49>: lea rdi,[rip+0x0] # 0xb1 <main+56>

0x00000000000000b1 <+56>: call 0xb6 <main+61>

0x00000000000000b6 <+61>: lea rsi,[rip+0x0] # 0xbd <main+68>

0x00000000000000bd <+68>: lea rdi,[rip+0x0] # 0xc4 <main+75>

0x00000000000000c4 <+75>: call 0xc9 <main+80>

0x00000000000000c9 <+80>: lea rsi,[rip+0x0] # 0xd0 <main+87>

0x00000000000000d0 <+87>: lea rdi,[rip+0x0] # 0xd7 <main+94>

0x00000000000000d7 <+94>: call 0xdc <main+99>

0x00000000000000dc <+99>: lea rsi,[rip+0x0] # 0xe3 <main+106>

0x00000000000000e3 <+106>: lea rdi,[rip+0x0] # 0xea <main+113>

0x00000000000000ea <+113>: call 0xef <main+118>

0x00000000000000ef <+118>: lea rsi,[rip+0x0] # 0xf6 <main+125>

0x00000000000000f6 <+125>: lea rdi,[rip+0x0] # 0xfd <main+132>

0x00000000000000fd <+132>: call 0x102 <main+137>

0x0000000000000102 <+137>: lea rsi,[rip+0x0] # 0x109 <main+144>

0x0000000000000109 <+144>: lea rdi,[rip+0x0] # 0x110 <main+151>

0x0000000000000110 <+151>: call 0x115 <main+156>

0x0000000000000115 <+156>: lea rax,[rbp-0x14]

0x0000000000000119 <+160>: mov rsi,rax

0x000000000000011c <+163>: lea rdi,[rip+0x0] # 0x123 <main+170>

0x0000000000000123 <+170>: call 0x128 <main+175>

0x0000000000000128 <+175>: mov eax,DWORD PTR [rbp-0x14]

0x000000000000012b <+178>: cmp eax,0x1

0x000000000000012e <+181>: jne 0x1c9 <main+336>

0x0000000000000134 <+187>: lea rax,[rbp-0x10]

0x0000000000000138 <+191>: mov rsi,rax

0x000000000000013b <+194>: lea rdi,[rip+0x0] # 0x142 <main+201>

0x0000000000000142 <+201>: call 0x147 <main+206>

0x0000000000000147 <+206>: mov rdx,rax

0x000000000000014a <+209>: lea rax,[rbp-0xc]

0x000000000000014e <+213>: mov rsi,rax

0x0000000000000151 <+216>: mov rdi,rdx

---Type <return> to continue, or q <return> to quit---

0x0000000000000154 <+219>: call 0x159 <main+224>

0x0000000000000159 <+224>: mov eax,DWORD PTR [rbp-0x10]

0x000000000000015c <+227>: mov esi,eax

0x000000000000015e <+229>: lea rdi,[rip+0x0] # 0x165 <main+236>

0x0000000000000165 <+236>: call 0x16a <main+241>

0x000000000000016a <+241>: lea rsi,[rip+0x0] # 0x171 <main+248>

0x0000000000000171 <+248>: mov rdi,rax

0x0000000000000174 <+251>: call 0x179 <main+256>

0x0000000000000179 <+256>: mov rdx,rax

0x000000000000017c <+259>: mov eax,DWORD PTR [rbp-0xc]

0x000000000000017f <+262>: mov esi,eax

0x0000000000000181 <+264>: mov rdi,rdx

0x0000000000000184 <+267>: call 0x189 <main+272>

0x0000000000000189 <+272>: lea rsi,[rip+0x0] # 0x190 <main+279>

0x0000000000000190 <+279>: mov rdi,rax

0x0000000000000193 <+282>: call 0x198 <main+287>

0x0000000000000198 <+287>: mov rcx,rax

0x000000000000019b <+290>: mov edx,DWORD PTR [rbp-0x10]

0x000000000000019e <+293>: mov eax,DWORD PTR [rbp-0xc]

0x00000000000001a1 <+296>: sub edx,eax

0x00000000000001a3 <+298>: mov eax,edx

0x00000000000001a5 <+300>: mov esi,eax

0x00000000000001a7 <+302>: mov rdi,rcx

0x00000000000001aa <+305>: call 0x1af <main+310>

0x00000000000001af <+310>: mov rdx,rax

0x00000000000001b2 <+313>: mov rax,QWORD PTR [rip+0x0] # 0x1b9 <main+320>

0x00000000000001b9 <+320>: mov rsi,rax

0x00000000000001bc <+323>: mov rdi,rdx

0x00000000000001bf <+326>: call 0x1c4 <main+331>

0x00000000000001c4 <+331>: jmp 0x97 <main+30>

0x00000000000001c9 <+336>: mov eax,DWORD PTR [rbp-0x14]

0x00000000000001cc <+339>: cmp eax,0x2

0x00000000000001cf <+342>: jne 0x268 <main+495>

0x00000000000001d5 <+348>: lea rax,[rbp-0x10]

0x00000000000001d9 <+352>: mov rsi,rax

0x00000000000001dc <+355>: lea rdi,[rip+0x0] # 0x1e3 <main+362>

0x00000000000001e3 <+362>: call 0x1e8 <main+367>

0x00000000000001e8 <+367>: mov rdx,rax

0x00000000000001eb <+370>: lea rax,[rbp-0xc]

0x00000000000001ef <+374>: mov rsi,rax

0x00000000000001f2 <+377>: mov rdi,rdx

0x00000000000001f5 <+380>: call 0x1fa <main+385>

0x00000000000001fa <+385>: mov eax,DWORD PTR [rbp-0x10]

0x00000000000001fd <+388>: mov esi,eax

---Type <return> to continue, or q <return> to quit---

0x00000000000001ff <+390>: lea rdi,[rip+0x0] # 0x206 <main+397>

0x0000000000000206 <+397>: call 0x20b <main+402>

0x000000000000020b <+402>: lea rsi,[rip+0x0] # 0x212 <main+409>

0x0000000000000212 <+409>: mov rdi,rax

0x0000000000000215 <+412>: call 0x21a <main+417>

0x000000000000021a <+417>: mov rdx,rax

0x000000000000021d <+420>: mov eax,DWORD PTR [rbp-0xc]

0x0000000000000220 <+423>: mov esi,eax

0x0000000000000222 <+425>: mov rdi,rdx

0x0000000000000225 <+428>: call 0x22a <main+433>

0x000000000000022a <+433>: lea rsi,[rip+0x0] # 0x231 <main+440>

0x0000000000000231 <+440>: mov rdi,rax

0x0000000000000234 <+443>: call 0x239 <main+448>

0x0000000000000239 <+448>: mov rcx,rax

0x000000000000023c <+451>: mov edx,DWORD PTR [rbp-0x10]

0x000000000000023f <+454>: mov eax,DWORD PTR [rbp-0xc]

0x0000000000000242 <+457>: add eax,edx

0x0000000000000244 <+459>: mov esi,eax

0x0000000000000246 <+461>: mov rdi,rcx

0x0000000000000249 <+464>: call 0x24e <main+469>

0x000000000000024e <+469>: mov rdx,rax

0x0000000000000251 <+472>: mov rax,QWORD PTR [rip+0x0] # 0x258 <main+479>

0x0000000000000258 <+479>: mov rsi,rax

0x000000000000025b <+482>: mov rdi,rdx

0x000000000000025e <+485>: call 0x263 <main+490>

0x0000000000000263 <+490>: jmp 0x97 <main+30>

0x0000000000000268 <+495>: mov eax,DWORD PTR [rbp-0x14]

0x000000000000026b <+498>: cmp eax,0x3

0x000000000000026e <+501>: jne 0x97 <main+30>

0x0000000000000274 <+507>: lea rax,[rbp-0x10]

0x0000000000000278 <+511>: mov rsi,rax

0x000000000000027b <+514>: lea rdi,[rip+0x0] # 0x282 <main+521>

0x0000000000000282 <+521>: call 0x287 <main+526>

0x0000000000000287 <+526>: mov rdx,rax

0x000000000000028a <+529>: lea rax,[rbp-0xc]

0x000000000000028e <+533>: mov rsi,rax

0x0000000000000291 <+536>: mov rdi,rdx

0x0000000000000294 <+539>: call 0x299 <main+544>

0x0000000000000299 <+544>: mov eax,DWORD PTR [rbp-0x10]

0x000000000000029c <+547>: mov esi,eax

0x000000000000029e <+549>: lea rdi,[rip+0x0] # 0x2a5 <main+556>

0x00000000000002a5 <+556>: call 0x2aa <main+561>

0x00000000000002aa <+561>: lea rsi,[rip+0x0] # 0x2b1 <main+568>

0x00000000000002b1 <+568>: mov rdi,rax

---Type <return> to continue, or q <return> to quit---

0x00000000000002b4 <+571>: call 0x2b9 <main+576>

0x00000000000002b9 <+576>: mov rdx,rax

0x00000000000002bc <+579>: mov eax,DWORD PTR [rbp-0xc]

0x00000000000002bf <+582>: mov esi,eax

0x00000000000002c1 <+584>: mov rdi,rdx

0x00000000000002c4 <+587>: call 0x2c9 <main+592>

0x00000000000002c9 <+592>: lea rsi,[rip+0x0] # 0x2d0 <main+599>

0x00000000000002d0 <+599>: mov rdi,rax

0x00000000000002d3 <+602>: call 0x2d8 <main+607>

0x00000000000002d8 <+607>: mov rcx,rax

0x00000000000002db <+610>: mov eax,DWORD PTR [rbp-0x10]

0x00000000000002de <+613>: mov esi,DWORD PTR [rbp-0xc]

0x00000000000002e1 <+616>: cdq

0x00000000000002e2 <+617>: idiv esi

0x00000000000002e4 <+619>: mov esi,eax

0x00000000000002e6 <+621>: mov rdi,rcx

0x00000000000002e9 <+624>: call 0x2ee <main+629>

0x00000000000002ee <+629>: mov rdx,rax

0x00000000000002f1 <+632>: mov rax,QWORD PTR [rip+0x0] # 0x2f8 <main+639>

0x00000000000002f8 <+639>: mov rsi,rax

0x00000000000002fb <+642>: mov rdi,rdx

0x00000000000002fe <+645>: call 0x303 <main+650>

0x0000000000000303 <+650>: jmp 0x97 <main+30>

0x0000000000000308 <+655>: mov eax,0x0

0x000000000000030d <+660>: mov rcx,QWORD PTR [rbp-0x8]

0x0000000000000311 <+664>: xor rcx,QWORD PTR fs:0x28

0x000000000000031a <+673>: je 0x321 <main+680>

0x000000000000031c <+675>: call 0x321 <main+680>

0x0000000000000321 <+680>: leave

0x0000000000000322 <+681>: ret

End of assembler dump.

Broken code

#include<iostream>

#include<string>

using namespace std;

int main() {

int v1 = 0;

while (v1 != 5) {

will only be 1, 2, 3 or 4 never 5cout <<

("----------------");

cout <<("- 1)Add -");

cout <<("- 2)Subtract -");

cout <<("- 3)Multiply -");

cout <<("- 4)Exit -");

cout <<("----------------");

//getting users selection

cout<< v1 << endl;

int v2;

int v3;

int v4;

int v5;

int v6;

int v7;

int v8;

int v9;

// v1 to v9 should all be doubles to avoid security issues and breaking the application

if (v1 != 1) {

if (v1 != 2) {

if (v1 == 3) {

cout << to\_string(v3) - to\_string(v2) << endl;

v9 = v3;

v4 = v2;

cout << to\_string(v9) - to\_string(v4) = v9/v4 << endl;

//this displays multiplication but is doingdivision

}

}

else {

cout << to\_string(v3) to\_string(v2)v7 = v3;

v8 = v2;

cout << to\_string(v7) - to\_string(v8) = v8 - v7 << endl;

//This shows subtraction and the users choice was subtraction but the numbers are being added

}

}

else {

cout << to\_string(v3) to\_string(v2)v5 = v3;

v6 = v2;

cout << to\_string(v5) - to\_string(v6) = v5 + v6 << endl;

//This displays addition and should be additionbut the operation is subtraction

}

}return 0;

}

Correct code

#include<iostream>

#include<string>

using namespace std;

string DisplayMenu(){

int choice = 0;

int x, y, z;

cout <<("- 1)Add -");

cout <<("- 2)Subtract -");

cout <<("- 3)Multiply -");

cout <<("- 4)Divide -");

cout <<("--5)Exit -");

cin >> choice;

if(choice == 1)

{

cin >> x;

cin >> y;

z = x + y;

cout <<"your answer is" << z << endl;

}

if(choice == 2)

{

cin >> x;

cin >> y;

z = x - y;

cout <<"your answer is" << z << endl;

}

if(choice == 3)

{

cin >> x;

cin >> y;

z = x \* y;

cout <<"your answer is" << z << endl;

}

if(choice == 4)

{

cin >> x;

cin >> y;

z = x / y;

cout <<"your answer is" << z << endl;

}

if(choice == 5)

{

cin >> x;

cin >> y;

z = x + y;

cout <<"GoodBye" << z << endl;

}

}

int main() {

int v;

if(v == 5){

}

return 0;

}