

Zawartość:

- zamiana wartościami dwóch zmiennych liczbowych bez użycia trzeciej zmiennej
- zamiana wartościami dwóch zmiennych tekstowych bez użycia trzeciej zmiennej

Obrócenie zmiennych liczbowych bez konieczności użycia trzeciej zmiennej.

```
int a = 10;
int b = 20;

a += b;    //a = 30
b = a - b; //b = 10
a = a - b; //a = 20
```

Każdy to zna. Prosta sprawa.

Czy da się w ten sam sposób obrócić stringi?

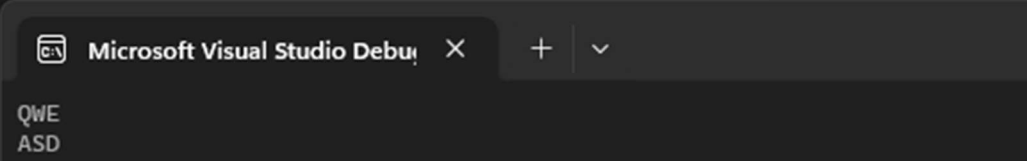
```
char a[4] = "ASD";
char b[4] = "QWE";

*(int*)a = *(int*)a + *(int*)b;

*(int*)b = *(int*)a - *(int*)b;

*(int*)a = *(int*)a - *(int*)b;

std::cout << a << std::endl;
std::cout << b << std::endl;
```



W pewnych specyficznych warunkach jak widać się da.

- każdy znak to liczba
- każdy znak to bajt
- cztery znaki to cztery bajty
- można na takim stringu (const char*) zrobić cast na (int*) i wyłuskać wartość
- dzięki czemu dostaniemy liczbę int składającą się z bajtów danego stringa
- liczby te dodajemy i odejmujemy
- gdybyśmy castowali na long long to możliwe było by zmienienie ciągów nawet 8 znakowych (włączając w to terminator) bo long long ma 8 bajtów
- powyższy kod zadziała dla stringów, które po dodaniu jako liczba nie zrobią przepełnienia (wynik dodawania musi mieścić się w czterech bajtach w przypadku inta)

A4	41	53	44	00	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	ASD.	EEEEEEEEEEEE
B2	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	EEEEEEEEEEEEEEEE
C0	cc	cc	cc	cc	51	57	45	00	cc	cc	cc	cc	cc	cc	cc	EEEQWE.	EEEEEE
CE	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	EEEEEEEEEEEEEEEE	
DC	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	f9	2d	EEEEEEEEEEEEEEû-	
EA	a4	06	fd	7f	00	00	50	ae	bb	06	fd	7f	00	00	00	«.ý...P»».ý...	

Tak stringi wyglądają w pamięci. Dodawana/odejmowana jest ich reprezentacja liczbowa.