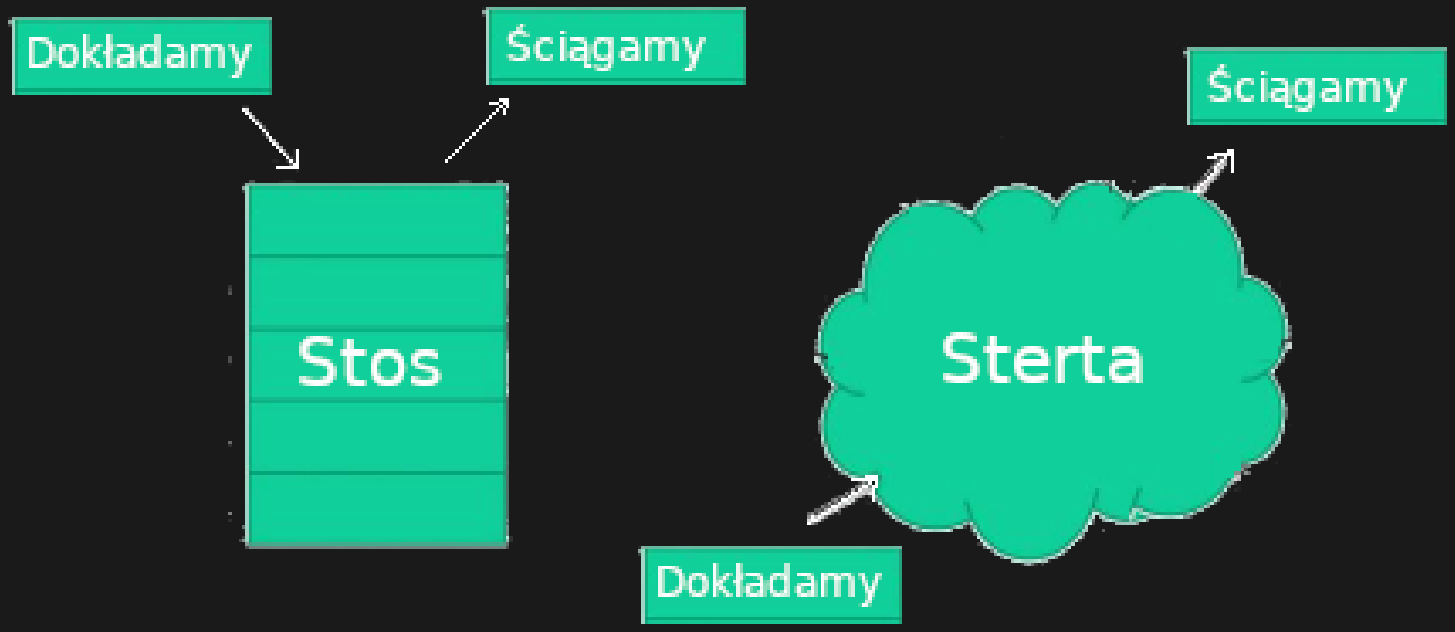
Stos i sterta

Adam Djellouli

April 19, 2020

Pamięć podzielona jest m.in. między stos i stertę.

Stos vs sterta



Stos

- stos to część pamięci przechowująca tymczasowe zmienne;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;
- łatwo wkładać i zdejmować;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;
- łatwo wkładać i zdejmować;
- kiedy zmienne nie są więcej wykorzystywane, ściągane są ze stosu;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;
- łatwo wkładać i zdejmować;
- kiedy zmienne nie są więcej wykorzystywane, ściągane są ze stosu;
- liniowa struktura danych;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;
- łatwo wkładać i zdejmować;
- kiedy zmienne nie są więcej wykorzystywane, ściągane są ze stosu;
- liniowa struktura danych;
- LIFO;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;
- łatwo wkładać i zdejmować;
- kiedy zmienne nie są więcej wykorzystywane, ściągane są ze stosu;
- liniowa struktura danych;
- LIFO;
- stos ma stałą wielkość określoną przez twój komputer;

Stos

- stos to część pamięci przechowująca tymczasowe zmienne;
- argumenty przekazywane funkcjom są zbierane na stosie;
- zmienne tworzone w obrębie funkcji;
- pamięć dostępna tylko lokalnie;
- łatwo wkładać i zdejmować;
- kiedy zmienne nie są więcej wykorzystywane, ściągane są ze stosu;
- liniowa struktura danych;
- LIFO;
- stos ma stałą wielkość określoną przez twój komputer;
- stack overflow gdy chcemy włożyć zbyt wiele na stos;
- popularny błąd: próba użycia zmiennej zapisanej na stosie poza funkcją, która tą zmienną odłożyła na stos;

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;
- dostęp do pamięci poprzez wskaźniki;

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;
- dostęp do pamięci poprzez wskaźniki;
- allokujemy (malloc) i dealokujemy (free);

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;
- dostęp do pamięci poprzez wskaźniki;
- allokujemy (malloc) i dealokujemy (free);
- w przeciwnym razie wyciek pamięci (pamięć której nie używamy jest dla nas trzymana);

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;
- dostęp do pamięci poprzez wskaźniki;
- allokujemy (malloc) i dealokujemy (free);
- w przeciwnym razie wyciek pamięci (pamięć której nie używamy jest dla nas trzymana);
- nie ma ograniczeń na stertę poza fizyczną pamięcią komputera;

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;
- dostęp do pamięci poprzez wskaźniki;
- allokujemy (malloc) i dealokujemy (free);
- w przeciwnym razie wyciek pamięci (pamięć której nie używamy jest dla nas trzymana);
- nie ma ograniczeń na stertę poza fizyczną pamięcią komputera;
- zmienne dostępne z całego programu;

Sterta

- duże pole pamięci, które można wykorzystywać dynamicznie;
- nikt za nas nie zarządza pamięcią;
- dostęp do pamięci poprzez wskaźniki;
- allokujemy (malloc) i dealokujemy (free);
- w przeciwnym razie wyciek pamięci (pamięć której nie używamy jest dla nas trzymana);
- nie ma ograniczeń na stertę poza fizyczną pamięcią komputera;
- zmienne dostępne z całego programu;
- wolniejsza w dostępie od stosu;

Kiedy wolno ci użyć sterty?

- tylko wtedy gdy nie można już zrobić nic innego;
- duże tablice i struktury;
- gdy chcesz żeby strutkury danych zmieniały swój rozmiar w trakcie trwania programu (rosły i malały);
- allokujemy (malloc) i dealokujemy (free);

```
1 #include <stdlib.h>
2
3 int N = 10;
4
5 //zarezerwuj pamiec dla N wartosci int
6 int *p =  (int*) malloc (N * sizeof(int));
7
8 //uwolnij pamiec
9 free(p);
```

Malloc kontra new

malloc	**new**
funkcja	operator
zwraca void*	zwraca różne typy
w razie błędu zwraca NULL	w razie błędu wyrzuca wyjątek
nie może być nadpisany	może być nadpisany
jest możliwość zmiany rozmiaru bufora	nie ma takiej możliwości
podajemy liczbę bajtów do alokacji	kompilator liczy za nas
tylko alokuje pamięć	jeszcze wywołuje konstruktor