Lab03 - podstawowe struktury danych: tablice

Mateusz Krawczuk, nr indeksu 209147

Wygenerowano przez Doxygen 1.8.6

Śr, 25 mar 2015 14:41:17

Część I

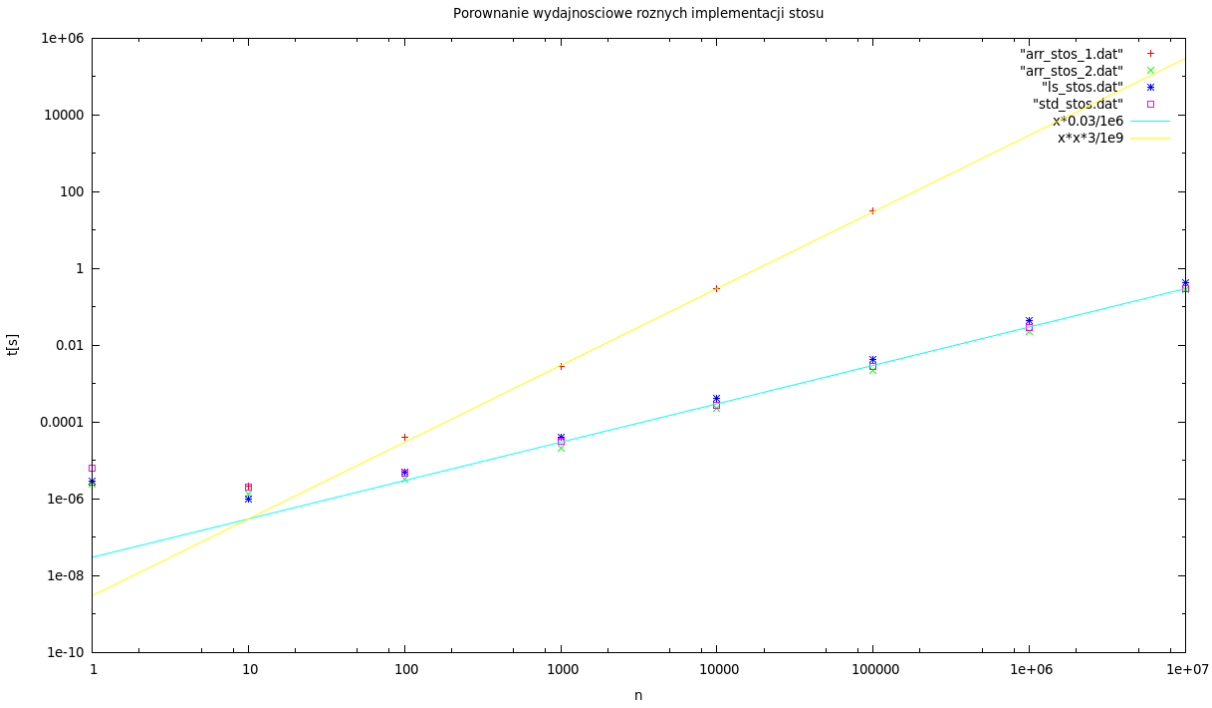
Streszczenie

Niniejszy dokument zawiera wyniki pomiaru czasu, którego potrzebował mój komputer na zapełnienie zaimplementowanych przeze mnie struktur danych opartych na typach tablicowych: stos, kolejka oraz lista zestawami danych o długościach od 1 do 1e7 elementów. Zawiera także dokumentację kodu, który pojawił się w projekcie od poprzedniego sprawozdania.

Część II

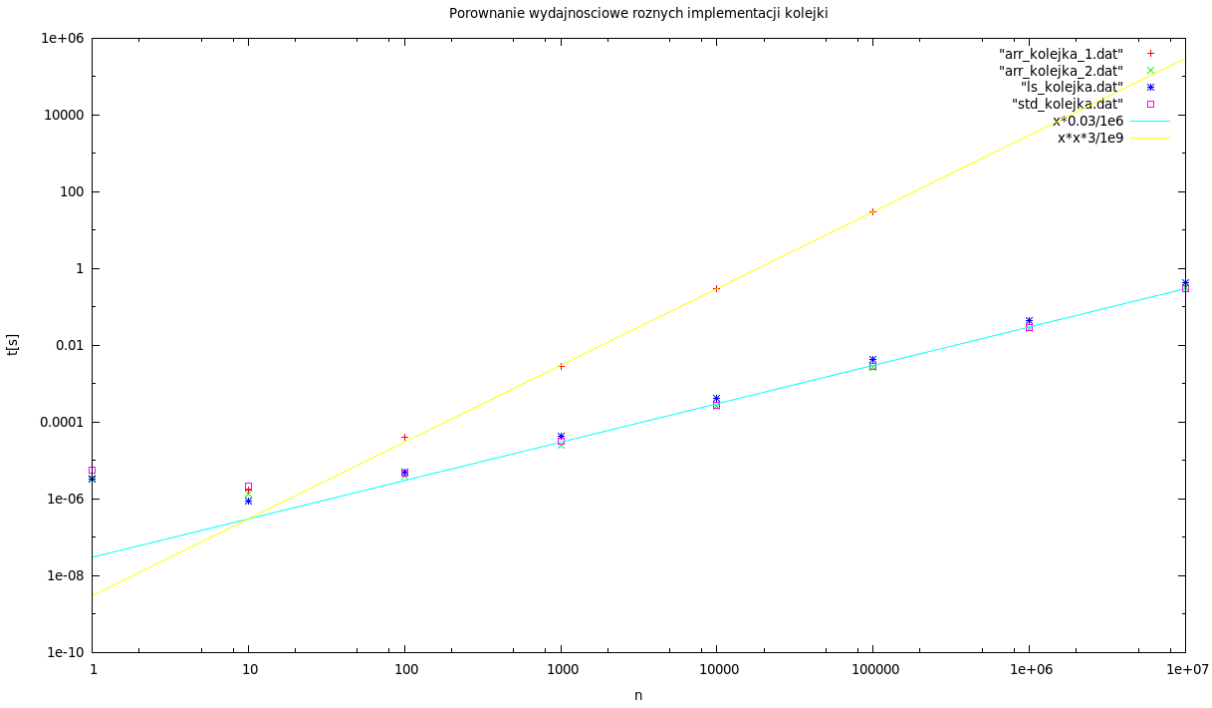
Sprawozdanie

Obliczenia wykonano na 64-bitowym procesorze AMD Athlon X2. Wszystkie implementacje zostały badane pod kątem najgorszego scenariusza. Wykresy przedstawiają porównania zależnośćiczasu wykonywania operacji od długości ciągu danych dla różnych implementacji poszczególnych struktur. Porównywane implementacje to:
- kontenery z biblioteki standardowej,
- implementacja z użyciem list jednokierunkowych,
- z użyciem tablic, gdzie pojemność tablicy jest zwiększana o jeden element,
- z użyciem tablic, gdzie pojemność tablicy zwiększana jest dwukrotnie.



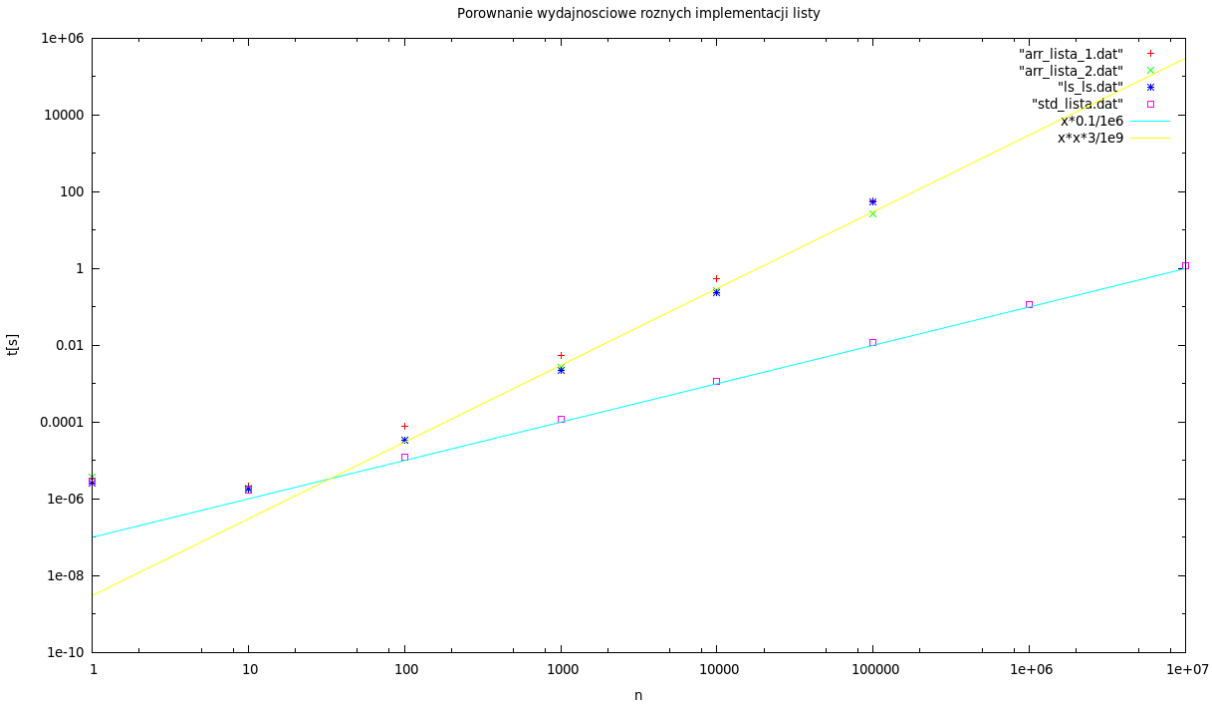
Wykres 1. Porównanie różnych implementacji stosu.
Zależność czasu wykonywania operacji od ilości danych wrzucanych na szczyt stosu dla wszystkich implementacji z wyjątkiem tablicowej-zwiększanej-o-1 dobrze aproksymuje linia prosta po czym można przypuszczać, że złożoność tej operacji jest $O(n)$. Dla stosu z biblioteki standardowej i zaimplementowanego za pomocą listy jest to trafny wniosek, gdyż ich złożoność jest faktycznie liniowa. Inaczej jest w przypadku stosu tablicowego-powiększanego-dwukrotnie, bowiem w trakcie powiększania go co jakiśczas następuje przepisanie całej jego zawartośći do innego miejsca w pamięci, co oczywiście jest operacją liniową. Jednak w związku z tym, że wraz ze zwiększaniem się ilości danych na stosie częstotliwość wystąpienia operacji przenoszenia maleje, wzrasta wydajność czasowa kosztem wydajności pamięciowej. Jak widać na wykresie, prosta bardzo dobrze aproksymuje stopieńzłożoności dla tej implementacji.

W przypadku stosu tablicowego-powiększanego-o-1 za każdym razem, gdy wyczerpiemy zapas pojemności kontenera, następuje realokacja całej jego zawartości do nowego miejsca z zapasem mogącym zmieścić jeden element - ten zapas wyczerpuje się w momencie, w którym wrzucimy na stos kolejny element. W konsekwencji wrzucenie na stos n danych kosztuje nas n-krotne n przepisań całego stosu, co razem daje n^2 operacji, zatem złożonośćobliczeniowa tego przedsięwzięcia wynosi $O(n^2)$.Gdyby wykres był w skali liniowej, linia aproksymująca współrzędne uzyskane z badań nad tą implementacją byłaby ramieniem paraboli.



Wykres 2. Porównanie różnych implementacji kolejki.

Sytuacja jest identyczna jak w przypadku stosu - ze względu na sposób implementacji wyniki są prawie identyczne. Różnice w tych strukturach byłyby widoczne, gdyby badano zdejmowanie z nich elementów.



Wykres 3. Porównanie różnych implmentacji listy.
Jedyną implementacją, w której umieszczanie elementów ma złożoność liniową jest lista z biblioteki standardowej. Jest to spodowane tym, że std::list wykorszytane do porównania jest listą dwukierunkową i nie jest wymagane każdorazowe przesuwanie jej zawartości podczas dodawania do niej elementu.

Złożoność obliczeniowa wszystkich moich implementacji jest $O(n^2)$, ponieważ napisana przez mnie lista jednokierunkowa została zaprojektowana w taki sposób, że dodawanie na jej początek nowego elementu (co jest najgorszym scenariuszem pod względem złożoności) wymaga operacji na każdym elemencie istniejącym w liście - jest to przesuwanie wskaźnika pomocniczego z końca na sam początek listy lub podnoszenie każdego elementu o jedno miejsce bliżej końca. Mamy więc dla n-elemenotwego zbioru danych n-krotne$n$ przesunięć wektora służące wpięciu nowego elementu listy na jej początek lub, w przypadku implementacji tablicowej, n-krotne n podniesień elementów bliżej końca by zrobić miejsce na nowy element w indeksie zerowym.

Część III

Dokumentacja przestrzeni nazw

1 Dokumentacja przestrzeni nazw arr

Komponenty

- class Kolejka
- class Lista
- class Stos

2 Dokumentacja przestrzeni nazw ls

Kolejka, abstrakcyjna struktura danych z buforem typu LIFO.

Komponenty

- class Kolejka
- class Kontener
- class Stos

2.1 Opis szczegółowy

Stos, abstrakcyjna struktura danych z buforem typu FIFO.

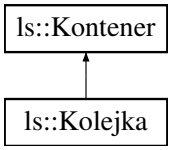
Część IV

Dokumentacja klas

3 Dokumentacja klasy ls::Kolejka

```
#include <kolejka.h>
```

Diagram dziedziczenia dla ls::Kolejka



Metody publiczne

- int pop ()

  Pop kolejki jaki jest, każdy widzi. Usuwa najstarszy element i zwraca wartość przez niego przechowywaną.

Dodatkowe Dziedziczone Składowe

3.1 Opis szczegółowy

Definicja w linii 14 pliku kolejka.h.

3.2 Dokumentacja funkcji składowych

3.2.1 int ls::Kolejka::pop ( ) [inline]

Definicja w linii 21 pliku kolejka.h.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- kolejka.h

4 Dokumentacja klasy arr::Kolejka

```
#include <kolejka_arr.h>
```

Metody publiczne

- Kolejka ()
  Konstruktor domyślny klasy arr::Kolejka.
- Kolejka (const Kolejka &lewy)
  Konstruktor kopiujący klasy arr::Kolejka.
- ~Kolejka ()
  Destruktor klasy arr::Kolejka.
- void push (int val)
  Wrzuca element na szczyt kolejki.
- int pop ()
  Zdejmuje najstarszy element z kolejki.
- bool empty ()
  Sprawdza, czy kolejka jest pusta.
- int size ()
  Zwraca rozmiar kolejki.

Metody prywatne

- void rozszerz_x2 ()
  Metoda zwiększa dwukrotnie pojemnosć kolejki.
- void rozszerz_1 ()
  Metoda zwiększa pojemnosć kolejki o 1.

Atrybuty prywatne

- int * tablica
- int rozmiar_kol
- int pojemnosc_kol

4.1 Opis szczegółowy

Definicja w linii 7 pliku kolejka_arr.h.

4.2 Dokumentacja konstruktora i destruktora

4.2.1 arr::Kolejka::Kolejka ( ) [inline]

Definicja w linii 26 pliku kolejka_arr.h.

4.2.2 arr::Kolejka::Kolejka (const Kolejka & *lewy*)

Bezużyteczny dla trzeciego zadania, ale na pewno kolega się ucieszy.

Definicja w linii 3 pliku kolejka_arr.cpp.

4.2.3 arr::Kolejka::∼Kolejka ( )

Definicja w linii 10 pliku kolejka_arr.cpp.

4.3 Dokumentacja funkcji składowych

4.3.1 bool arr::Kolejka::empty ( )

Zwraca

Zwraca prawdę, jeżeli kolejka jest pusta. W przeciwnym razie zwraca false.

Definicja w linii 44 pliku kolejka_arr.cpp.

4.3.2 int arr::Kolejka::pop ( )

Zwraca

Zwraca wartość przechowywaną w zdejmowanej cegiełce

Definicja w linii 32 pliku kolejka_arr.cpp.

4.3.3 void arr::Kolejka::push (int *val*)

Definicja w linii 15 pliku kolejka_arr.cpp.

4.3.4 void arr::Kolejka::rozszerz_1 ( ) [private]

Definicja w linii 64 pliku kolejka_arr.cpp.

4.3.5 void arr::Kolejka::rozszerz_x2 ( ) [private]

Definicja w linii 49 pliku kolejka_arr.cpp.

4.3.6 int arr::Kolejka::size ( ) [inline]

Definicja w linii 56 pliku kolejka_arr.h.

4.4 Dokumentacja atrybutów składowych

4.4.1 int arr::Kolejka::pojemnosc_kol [private]

Definicja w linii 12 pliku kolejka_arr.h.

4.4.2 int arr::Kolejka::rozmiar_kol [private]

Definicja w linii 11 pliku kolejka_arr.h.

4.4.3 int* arr::Kolejka::tablica [private]

Definicja w linii 10 pliku kolejka_arr.h.

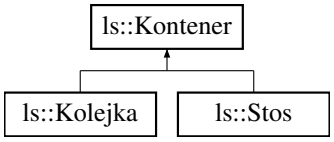
Dokumentacja dla tej klasy została wygenerowana z plików:

- kolejka_arr.h
- kolejka_arr.cpp

5 Dokumentacja klasy ls::Kontener

`#include <kontener.h>`

Diagram dziedziczenia dla ls::Kontener



Metody publiczne

- ~Kontener ()

  Konstruktor klasy Kontener.
- Kontener ()
- void push (int)

  Wrzuca nową cegiełkę na początek kontenera.
- void insert (int wartosc, int indeks)

  Umieszcza cegłę z wartością 'wartosc' w miejscu oddalonym o 'indeks' miejsc od początku kontenera.
- int size ()

  Liczy z ilu cegiełek składa się kontener.
- int erase (int)

  Usuwa z listy element o wybranym indeksie.
- int find (int)

  Odnajduje w kontenerze przekazaną w argumencie wartość.
- void show ()

  Wypisuje elementy listy od najmłodszego zaczynając.

Atrybuty chronione

- cegla * alfa

5.1 Opis szczegółowy

Definicja w linii 14 pliku kontener.h.

5.2 Dokumentacja konstruktora i destruktora

5.2.1 ls::Kontener::∼Kontener ( )

Konstruktor klasy Kontener inicjalizuje wskaźnik 'alfa' wartością NULL.

Definicja w linii 7 pliku kontener.cpp.

5.2.2 ls::Kontener::Kontener ( ) [inline]

Definicja w linii 27 pliku kontener.h.

5.3 Dokumentacja funkcji składowych

5.3.1 int ls::Kontener::erase (int *indeks*)

Zainicjalizowane są dwa wskaźniki na najmłodszą cegiełkę. Jeden jest ustawiany na element do usunięcia, drugi na element o jeden młodszy. Następuje roszada wskaźników: wskaźnik młodszej cegły wskazuje na cegłę starszą od usuwanej, zostaje zapisana wartość przechowywana przez usuwaną cegłę i wreszcie zwolniona zostaje pamięć zajmowana dotychczas przez cegłę.

Przykład 1: Wyrażenie obj.erase(0) usuwa najmłodszy element kontenera. Przykład 2: Wyrażenie obj.erase(obj.-size() - 1) usuwa najstarszy element. Przykład 3: Wyrażenie obj.erase(obj.size()) nie zmienia kontenera.

Zwraca

Zwraca wartość przechowywaną w usuniętej cegiełce.

Definicja w linii 65 pliku kontener.cpp.

5.3.2 int ls::Kontener::find (int *wartosc*)

Powołany do życia jest szpieg - wskaźnik na obiekt typu 'cegla', który przemierza kontener w poszukiwaniu najwcześniejszego wystąpienia poszukiwanej wartości. Operacji towarzyszy licznik, który śledzi ilość miniętych przez wskaźnik cegieł, którą metoda zwraca. Należy ją interpretować jako indeks cegły, gdzie najwcześniej wystąpiła poszukiwana wartość.

Zwraca

Zwraca indeks (liczony od 0 od najmłodszej cegiełki) najbliższego wystąpienia poszukiwanej wartości.

Definicja w linii 99 pliku kontener.cpp.

5.3.3 void ls::Kontener::insert (int *wartosc,* int *indeks*)

UWAGA: Indeks liczony jest od zera, od najmłodszej cegły.

Przykład 1: Wyrażenie obj.insert(5,0) jest równoważne wyrażeniu obj.push(5)

- element zawierający wartość 5 jest teraz najmłodszym elementem. Przykład 2: Wyrażenie obj.insert(3,obj.-size()) czyni element zawierający wartosć 3 najstarszym elementem.

Definicja w linii 21 pliku kontener.cpp.

5.3.4 void ls::Kontener::push (int *wart*)

Definicja w linii 13 pliku kontener.cpp.

5.3.5 void ls::Kontener::show ( )

Definicja w linii 116 pliku kontener.cpp.

5.3.6 int ls::Kontener::size ( )

Zainicjalizowany zostaje wskaźnik na strukturę 'cegla' wskazujący na najmłodszą cegiełkę. Zostaje on potem wysłany w epicką podróż na sam koniec kontenera (czyli do napotkania NULLa). Towarzyszy mu licznik, który zlicza mijane po drodze cegiełki, których ilosć funkcja zwraca.

Zwraca

Zwraca wielkość kontenera.

Definicja w linii 50 pliku kontener.cpp.

5.4 Dokumentacja atrybutów składowych

5.4.1 cegla* ls::Kontener::alfa [protected]

Definicja w linii 17 pliku kontener.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- kontener.h
- kontener.cpp

6 Dokumentacja klasy arr::Lista

`#include <lista_arr.h>`

Metody publiczne

- Lista ()
  Konstruktor domyślny klasy arr::Lista.
- Lista (const Lista &lewy)
  Konstruktor kopiujący klasy arr::Lista.
- ~Lista ()
  Destruktor klasy arr::Lista.
- int erase (int)
  Usuwa z listy element o wybranym indeksie.
- void insert (int, int)
  Umieszcza cegłę z wartością 'wartosc' w miejscu oddalonym o 'indeks' miejsc od początku kontenera.
- bool empty ()
  Sprawdza, czy lista jest pusta.
- int size ()
  Zwraca rozmiar listy.

Metody prywatne

- void rozszerz_x2 ()

  Metoda zwiększa dwukrotnie pojemnosć listy.
- void rozszerz_1 ()

  Metoda zwiększa pojemnosć listy o 1.

Atrybuty prywatne

- int * tablica
- int rozmiar_ls
- int pojemnosc_ls

6.1 Opis szczegółowy

Definicja w linii 6 pliku lista_arr.h.

6.2 Dokumentacja konstruktora i destruktora

6.2.1 arr::Lista::Lista ( ) [inline]

Definicja w linii 25 pliku lista_arr.h.

6.2.2 arr::Lista::Lista (const Lista & *lewy*)

Bezużyteczny dla trzeciego zadania, ale na pewno kolega się ucieszy.

6.2.3 arr::Lista::~Lista ( ) [inline]

Definicja w linii 35 pliku lista_arr.h.

6.3 Dokumentacja funkcji składowych

6.3.1 bool arr::Lista::empty ( )

Zwraca

Zwraca prawdę, jeżeli lista jest pusta. W przeciwnym razie zwraca false.

Definicja w linii 3 pliku lista_arr.cpp.

6.3.2 int arr::Lista::erase (int *indeks*)

Zapisuje wartość przechowywaną pod danym indeksem a potem przesuwa całą zawartość tablicy młodszą od danego elementu o 1 w stronę starszych elementów. Zmniejsza licznik rozmiar_ls o 1.

Przykład 1: Wyrażenie obj.erase(0) usuwa najmłodszy element kontenera. Przykład 2: Wyrażenie obj.erase(obj.-size() - 1) usuwa najstarszy element. Przykład 3: Wyrażenie obj.erase(obj.size()) nie zmienia kontenera.

Zwraca

Zwraca wartość przechowywaną w usuniętej cegiełce.

Definicja w linii 8 pliku lista_arr.cpp.

6.3.3 void arr::Lista::insert (int *value,* int *indeks*)

Zwiększa licznik rozmiar_ls o 1, odsuwa wszystkie elementy młodsze od danego indeksu o 1 od starszych i umieszcza wartość w odpowiednim indeksie. Może powodować realokację zawartości listy!

UWAGA: Indeks liczony jest od zera, od najmłodszej cegły.

Przykład 1: Wyrażenie obj.insert(5,0) jest równoważne wyrażeniu obj.push(5)

- element zawierający wartość 5 jest teraz najmłodszym elementem. Przykład 2: Wyrażenie obj.insert(3,obj.-size()) czyni element zawierający wartosć 3 najstarszym elementem.

Definicja w linii 20 pliku lista_arr.cpp.

6.3.4 void arr::Lista::rozszerz_1 ( ) [private]

Definicja w linii 59 pliku lista_arr.cpp.

6.3.5 void arr::Lista::rozszerz_x2 ( ) [private]

Definicja w linii 44 pliku lista_arr.cpp.

6.3.6 int arr::Lista::size ( ) [inline]

Definicja w linii 83 pliku lista_arr.h.

6.4 Dokumentacja atrybutów składowych

6.4.1 int arr::Lista::pojemnosc_ls [private]

Definicja w linii 11 pliku lista_arr.h.

6.4.2 int arr::Lista::rozmiar_ls [private]

Definicja w linii 10 pliku lista_arr.h.

6.4.3 int* arr::Lista::tablica [private]

Definicja w linii 9 pliku lista_arr.h.

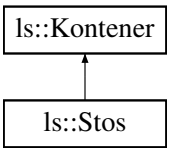
Dokumentacja dla tej klasy została wygenerowana z plików:

- lista_arr.h
- lista_arr.cpp

7 Dokumentacja klasy ls::Stos

`#include <stos.h>`

Diagram dziedziczenia dla ls::Stos



Metody publiczne

- ~Stos ()
- int pop ()

  Pop stosu jaki jest, każdy widzi. Zdejmuje najmłdoszy element i zwraca wartość przez niego przechowywaną.

Dodatkowe Dziedziczone Składowe

7.1 Opis szczegółowy

Definicja w linii 14 pliku stos.h.

7.2 Dokumentacja konstruktora i destruktora

7.2.1 ls::Stos::~Stos ( ) [inline]

Definicja w linii 17 pliku stos.h.

7.3 Dokumentacja funkcji składowych

7.3.1 int ls::Stos::pop ( ) [inline]

Definicja w linii 25 pliku stos.h.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- stos.h

8 Dokumentacja klasy arr::Stos

```
#include <stos_arr.h>
```

Metody publiczne

- Stos ()

  Konstruktor domyślny klasy Stos.
- Stos (const Stos &lewy)

  Konstruktor kopiujący klasy Stos.
- ~Stos ()

  Destruktor klasy Stos.
- void push (int val)

  Wrzuca element na szczyt stosu.
- int pop ()

  Zdejmuje najmłodszy element ze stosu.

- bool empty ()

  Sprawdza, czy stos jest pusty.
- int size ()

  Zwraca rozmiar stosu.

Metody prywatne

- void rozszerz_x2 ()

  Metoda zwiększa dwukrotnie pojemnosć stosu.
- void rozszerz_1 ()

  Metoda zwiększa pojemnosć stosu o 1.

Atrybuty prywatne

- int * tablica
- int rozmiar_stosu
- int pojemnosc_stosu

8.1 Opis szczegółowy

Definicja w linii 7 pliku stos_arr.h.

8.2 Dokumentacja konstruktora i destruktora

8.2.1 arr::Stos::Stos ( ) [inline]

Definicja w linii 26 pliku stos_arr.h.

8.2.2 arr::Stos::Stos (const Stos & *lewy*)

Bezużyteczny dla trzeciego zadania, ale na pewno kolega się ucieszy.

Definicja w linii 3 pliku stos_arr.cpp.

8.2.3 arr::Stos::~Stos ( )

Definicja w linii 10 pliku stos_arr.cpp.

8.3 Dokumentacja funkcji składowych

8.3.1 bool arr::Stos::empty ( )

Zwraca

Zwraca prawdę, jeżeli kolejka jest pusta. W przeciwnym razie zwraca false.

Definicja w linii 38 pliku stos_arr.cpp.

8.3.2 int arr::Stos::pop ( )

Definicja w linii 32 pliku stos_arr.cpp.

8.3.3 void arr::Stos::push (int *val*)

Definicja w linii 15 pliku stos_arr.cpp.

8.3.4 void arr::Stos::rozszerz_1 ( ) [private]

Definicja w linii 58 pliku stos_arr.cpp.

8.3.5 void arr::Stos::rozszerz_x2 ( ) [private]

Definicja w linii 43 pliku stos_arr.cpp.

8.3.6 int arr::Stos::size ( ) [inline]

Definicja w linii 54 pliku stos_arr.h.

8.4 Dokumentacja atrybutów składowych

8.4.1 int arr::Stos::pojemnosc_stosu [private]

Definicja w linii 12 pliku stos_arr.h.

8.4.2 int arr::Stos::rozmiar_stosu [private]

Definicja w linii 11 pliku stos_arr.h.

8.4.3 int* arr::Stos::tablica [private]

Definicja w linii 10 pliku stos_arr.h.

Dokumentacja dla tej klasy została wygenerowana z plików:

- stos_arr.h
- stos_arr.cpp