Zadanie 4.10

Wymagane

4. Dynamiczna alokacja pamięci I

10. Bufor cykliczny

Termin	Trudność	Próba	Ocena maszyny	Inspekcja kodu	Plagiat	Raporty
2022-06-16	★★★★☆	1				Raport główny
został 7 dni	Punkty: 4		Czas testu: 100sek			

Napisz program, który pozwoli użytkownikowi na wykonywanie podstawowych operacji na buforze cyklicznym.

W tym celu prygotuj strukturę `circular_buffer_t`, która będzie działała jak bufor cykliczny (wersja EN). Nowe elementy są dodawane na końcu tablicy i zmieniany jest indeks `end`, wskazujący pierwszą wolną pozycję. Po pobraniu elementu zmieniany jest indeks `begin` (w przypadku użycia funkcji `pop_front` - pobranie z początku bufora) lub `end` (w przypadku użycia funkcji `pop_back` - pobranie z końca bufora).

Struktura wspomagająca taką strukturę powinna zostać umieszczona w pliku nagłówkowym `circular_buffer.h` i wyglądać następująco:

```
struct circular_buffer_t {
    int *ptr;
    int begin;
    int end;
    int capacity;
    unsigned char full: 1;
};
```

gdzie:

- `ptr` - wskaźnik na tablicę, w której przechowywane będą dane (liczby całkowite),
- `begin` - indeks pierwszego elementu tablicy (zajętej pozycji),
- `end` - indeks pierwszej wolnej pozycji w tablicy, znajdującej się za ostatnim elementem,
- `capacity` - pojemność tablicy,
- `full` - flaga oznaczająca czy bufor jest pełny.

Zasada działania **bufora cyklicznego** jest następująca:

- Nowy element dodawany jest na "końcu" bufora `ptr`, pod wskaźnikiem `end`. Następnie wskaźnik ten jest zwiększany.
- Stary element pobierany jest z początku bufora `ptr` danego wskaźnikiem `begin`. Następnie wskaźnik ten jest zwiększany.
- W przypadku, gdy wskaźnik `begin` lub `end` przekroczy długość bufora `ptr`, jest zawiajny na pozycję `0`.

Przygotuje funkcje o następujących prototypach, deklaracje funkcji umieść w pliku nagłówkowym `circular_buffer.h`, a definicje w pliku `circular_buffer.c`:

```
int circular_buffer_create(struct circular_buffer_t *a, int N);
int circular_buffer_create_struct(struct circular_buffer_t **cb, int N);

void circular_buffer_destroy(struct circular_buffer_t *a);
void circular_buffer_destroy_struct(struct circular_buffer_t **a);

int circular_buffer_push_back(struct circular_buffer_t *cb, int value);
int circular_buffer_pop_front(struct circular_buffer_t *a, int *err_code);
int circular_buffer_pop_back(struct circular_buffer_t *a, int *err_code);

int circular_buffer_empty(const struct circular_buffer_t *a);
int circular_buffer_full(const struct circular_buffer_t *a);

void circular_buffer_display(const struct circular_buffer_t *a);
```

```
int circular_buffer_create(struct circular_buffer_t *cb, int N);
```

Funkcja alokuje pamięć na tablicę `N` elementów w buforze `cb` i ustawia odpowiednie wartości poszczególnych pól tej struktury. *Wartosc zwracana*:

- `1` - w przypadku błędnych danych wejściowych,
- `2` - jeżeli alokacja pamięci nie powiodła się,
- `0` - w przypadku sukcesu.

```
int circular_buffer_create_struct(struct circular_buffer_t **cb, int N);
```

Funkcja alokuje pamięć na strukturę `cb`, a następnie na tablicę `N` elementów w buforze `cb` i ustawia odpowiednie wartości poszczególnych pól tej struktury. *Wartosc zwracana*:

- `1` - w przypadku błędnych danych wejściowych lub braku ich spójności,
- `2` - jeżeli alokacja pamięci nie powiodła się,
- `0` - w przypadku sukcesu.

W przypadku niepowodzenia funkcja powinna zwolnić całą zaalokowaną pamięć.

```
void circular_buffer_destroy(struct circular_buffer_t *cb);
```

Funkcja zwalnia pamięć przydzieloną w strukturze `cb`.

```
void circular_buffer_destroy_struct(struct circular_buffer_t **cb);
```

Funkcja zwalnia pamięć przydzieloną na strukturę `cb` oraz w strukturze `cb`.

```
int circular_buffer_push_back(struct circular_buffer_t *cb, int value);
```

Funkcja dodaje element `value` na koniec tablicy w strukturze `cb`, aktualizując wartości odpowiednich pól struktury.

Wartość zwracana:

- `1` - w przypadku błędnych danych wejściowych lub braku ich spójności,
- `0` - w przypadku poprawnego dodania elementu do tablicy.

W przypadku próby dodania elementy do bufora, który jest już pełny, funkcja powinna nadpisać najwcześniej dodany element.

```
int circular_buffer_pop_front(struct circular_buffer_t *cb, int *err_code);
```

Funkcja zwraca i usuwa **pierwszy** element tablicy w strukturze `cb`, aktualizując wartości odpowiednich pól struktury.

Funkcja ustawia kod błędu `err_code` (jeśli to możliwe) na:

- `1` - w przypadku błędnych danych wejściowych lub braku ich spójności,
- `2` - w przypadku kiedy bufor jest pusty lub
- `0` - w przypadku poprawnego pobrania elementu z tablicy.

Wartość zwracana:

- wartość usuniętego elementu z początku bufora (jesli kod błędu jest `0`) lub
- wartość nieokreślona, jeśli kod błędu jest różny od `0`.

```
int circular_buffer_pop_back(struct circular_buffer_t *cb, int *err_code);
```

Funkcja zwraca i usuwa **ostatni** element tablicy w strukturze `cb`, aktualizując wartości odpowiednich pól struktury.

Funkcja ustawia kod błędu `err_code` (jeśli to możliwe) na:

- `1` - w przypadku błędnych danych wejściowych lub braku ich spójności,
- `2` - w przypadku kiedy bufor jest pusty lub

- `0` - w przypadku poprawnego pobrania elementu z tablicy.

Wartość zwracana:

- wartość usuniętego elementu z końca bufora (jesli kod błędu jest `0`) lub
- wartość nieokreślona, jeśli kod błędu jest różny od `0`.

```
int circular_buffer_empty(const struct circular_buffer_t *cb);
```

Funkcja sprawdza czy bufor cykliczny `cb` jest pusty.

Wartość zwracana:

- `-1` - w przypadku błędnych danych wejściowych lub braku ich spójności,
- `1` - jeżeli w buforze `cb` nie ma żadnych danych,
- `0` - jeżeli bufor nie jest pusty.

```
int circular_buffer_full(const struct circular_buffer_t *cb);
```

Funkcja sprawdza czy bufor cykliczny `cb` jest pełny.

Wartość zwracana:

- `-1` - w przypadku błędnych danych wejściowych lub braku ich spójności,
- `1` - jeżeli w buforze `cb` jest pełny,
- `0` - jeżeli bufor nie jest pełny.

```
void circular_buffer_display(const struct circular_buffer_t *cb);
```

Funkcja wyświetla zawartość bufora cyklicznego `cb` w jednej linii, z wartościami oddzielonymi spacjami. W przypadku gdy bufor jest pusty lub struktura przekazana do funkcji jest nieprawidłowa, funkcja `display` nie podejmuje żadnej akcji.

Podczas wyświetlania należy pamiętać, iż bufor cykliczny ma swój początek `begin` oraz koniec `end`.

Napisz program, który pozwoli użytkownikowi na wykonywanie podstawowych operacji na buforze cyklicznym.

Na początek program powinien zapytać użytkownika o pojemność bufora oraz utworzyć bufor o zadanej wielkości.

- W przypadku kiedy użytkownik wprowadzi nieprawidłowe znaki program powinien wyświetlić komunikat `Incorrect input` i zakończyć działanie z kodem błędu `1`.
- W przypadku podania błędnych danych program powinien wyświetlić komunikat `Incorrect input data` i zakończyć działanie z kodem błędu `2`.
- Jeżeli nie uda się zaalokować żądanego obszaru pamięci program powinien wyświetlić komunikat `Failed to allocate memory` i zwrócić kod błędu `8`.

Jeżeli udało się utworzyć bufor, program powinien w pętli pytać użytkownika o wybór operacji:

- `0` - Zakończenie działania programu.
- `1` - Dodanie elementu do tablicy. Program pobiera od użytkownika wartość, która ma zostać dodana do bufora.
- `2` - Pobranie ostatnio dodanego elementu do bufora. Program powinien wyświetlić wartość pobranego elementu. Jeżeli bufor jest *pusty* to program powinien wyświetlić komunikat `Buffer is empty`.
- `3` - Pobranie najwcześniej dodanego elementu do bufora. Program powinien wyświetlić wartość pobranego elementu. Jeżeli bufor jest *pusty* to program powinien wyświetlić komunikat `Buffer is empty`.
- `4` - Wyświetlanie całej zawartości bufora. W przypadku kiedy bufor jest pusty program powinien wyświetlić komunikat `Buffer is empty`
- `5` - sprawdzanie czy bufor jest pusty. Program powinien wypisać:
  - `1` jeżeli bufor jest pusty lub
  - `0` w przeciwnym przypadku,
- `6` - sprawdzanie czy bufor jest pełny. Program powinien wypisać:

    - `1` jeżeli bufor jest pełny lub
    - `0` w przeciwnym przypadku.
- W przypadku podania innej wartości operacji program powinien wyświetlić komunikat `Incorrect input data` i kontynuować działanie.

- W przypadku wprowadzenia nieprawidłowych znaków program powinien wyświetlić komunikat `Incorrect input` i zakończyć działanie z kodem błędu `1`.

Przykładowa interakcja z programem -- sukces:

```
Podaj rozmiar bufora: 9⏎
Co chcesz zrobic? 2⏎
Buffer is empty⏎
Co chcesz zrobic? 3⏎
Buffer is empty⏎
Co chcesz zrobic? 30⏎
Incorrect input data⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 1⏎
Co chcesz zrobic? 5⏎
0⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -10⏎
Co chcesz zrobic? 2⏎
-10⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -9⏎
Co chcesz zrobic? 4⏎
1 -9 ⏎
Co chcesz zrobic? 2⏎
-9⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 1⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 5⏎
Co chcesz zrobic? 5⏎
0⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 8⏎
Co chcesz zrobic? 5⏎
0⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -6⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -2⏎
Co chcesz zrobic? 2⏎
-2⏎
Co chcesz zrobic? 4⏎
1 1 5 8 -6 ⏎
Co chcesz zrobic? 3⏎
1⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -7⏎
Co chcesz zrobic? 4⏎
1 5 8 -6 -7 ⏎
Co chcesz zrobic? 5⏎
0⏎
Co chcesz zrobic? 4⏎
1 5 8 -6 -7 ⏎
Co chcesz zrobic? 3⏎
1⏎
Co chcesz zrobic? 3⏎
5⏎
Co chcesz zrobic? 2⏎
-7⏎
Co chcesz zrobic? 4⏎
8 -6 ⏎
Co chcesz zrobic? 4⏎
8 -6 ⏎
Co chcesz zrobic? 5⏎
0⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -4⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -2⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 2⏎
Co chcesz zrobic? 2⏎
2⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -5⏎
```

```
Co chcesz zrobic? 6⏎
0⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 3⏎
Co chcesz zrobic? 4⏎
8 -6 -4 -2 -5 3 ⏎
Co chcesz zrobic? 6⏎
0⏎
Co chcesz zrobic? 1⏎
Podaj liczbe 8⏎
Co chcesz zrobic? 0⏎
```

Przykładowa interakcja z programem -- brak pamięci:

Limit sterty: 12 bajtów

```
Podaj rozmiar bufora: 7⏎
Failed to allocate memory⏎
```

Przykładowa interakcja z programem -- błąd danych:

```
Podaj rozmiar bufora: 0⏎
Incorrect input data
```

```
Podaj rozmiar bufora: -649⏎
Incorrect input data
```

```
Podaj rozmiar bufora: 13⏎
Co chcesz zrobic? 1⏎
Podaj liczbe -96⏎
Co chcesz zrobic? uLaUl⏎
Incorrect input
```

Uwagi

- W programie nie wolno deklarować zmiennej typu `struct circular_buffer_t`, zamiast tego zadeklaruj wskaźnik na strukturę.
- W programie nie wolno używać operatora `[]`!
- Definicję struktury oraz deklaracje funkcji umieść w pliku nagłówkowym `circular_buffer.h`, a definicje funkcji w pliku `circular_buffer.c`.
