ProgramowanieProceduralne

IS_L9

```
void qsort(void * base,size_t num, size_t width,int( * compare )( const void *, const void * ));
```

- `const void *base` - wskaźnik na tablicę, która ma zostać posortowana.
- `size_t num` - liczba elementów w tablicy.
- `size_t width` - liczba bajtów zajmowanych przez jeden element tablicy.
- `int (*compare ) ( const void *, const void *)` - funkcja porównująca elementy tablicy.

  Do argumentów przedmiotowej funkcji trafiają wskaźniki na elementy obecnie porównywane.
  Funkcja przekazana jako argument `compare` powinna zwracać następujące wartości:
  - `< 0` - gdy wartość argumentu pierwszego jest mniejsza od argumentu drugiego;
  - `= 0` - gdy wartość argumentu pierwszego jest równa wartości argumentu drugiego;
  - `> 0` - gdy wartość argumentu pierwszego jest większa od argumentu drugiego.

```
void *bsearch(const void *key, const void *base, size t nmemb, size t size, int (*compare)(const void *, const void *));
```

Funkcja zwraca wskaźnik na element tablicy `base`, pasujący do szukanego klucza `key`. Jeżeli klucz nie został znaleziony, funkcja zwraca wartość `NULL`.
Zawartość tablicy powinna być posortowana w kolejności rosnącej zgodnie z funkcją porównawczą wykorzystywaną przez `compare`.
Jeżeli tablica zawiera elementy posiadające zduplikowane szukane klucze to nie wiadomo, adres którego z nich zostanie zwrócony.

- `const void *key` - poszukiwany klucz
- `const void *base` - wskaźnik na posortowaną tablicę, która ma zostać przeszukana.
- `size_t num` - liczba elementów w tablicy.
- `size_t width` - liczba bajtów zajmowanych przez jeden element tablicy.
- `int (*compare ) ( const void *, const void *)` - funkcja porównująca elementy tablicy.

  Do pierwszego argumentu przedmiotowej funkcji trafia wskaźnik na element poszukiwany key, natomiast do drugiego wskaźnik na element tablicy base z którym ma nastąpić porównanie.
  Funkcja przekazana jako argument `compare` powinna zwracać następujące wartości:
  - `< 0` - gdy wartość argumentu pierwszego jest mniejsza od argumentu drugiego;
  - `= 0` - gdy wartość argumentu pierwszego jest równa wartości argumentu drugiego;
  - `> 0` - gdy wartość argumentu pierwszego jest większa od argumentu drugiego.

1.(3) Proszę napisać program, w którym zostanie utworzona i wypisana wierszami, tablica 20 łańcuchów, każdy o długości 15 znaków. Łańcuchy należy wypełnić losowymi literami - dużymi i małymi. Następnie korzystając z funkcji `qsort` proszę posortować **każdy** wiersz według następującej zasady :

- według "wartości bezwzględnej " znaku
- w obrębie jednej litery mała litera przed dużą

Po posortowaniu każdego wiersza trzeba posortować całą tablicę łańcuchów tak, aby wiersze były ułożone alfabetycznie.

Na koniec proszę wypisać tablicę wierszami.

```
Przyklad :
Wygenerowane łańcuchy:
      ajiAdfTyahiKloU
      zsiAtfTdkHfKlol
      bcBhyTfBhGjHyLU
```

```
Posortowany każdy łańcuch:
      aaAdfhiijKloTUy
      AdffHikKllostTz
      bBBcfGhhHjLTUyy
```

```
Posortowana tablica łańcuchów:
      AdffHikKllostTz
      aaAdfhiijKloTUy
      bBBcfGhhHjLTUyy
```

2.(3) Proszę napisać program, który przyjmuje jako **argument wywołania** łańcuch znaków oraz liczbę typu double. W programie mamy dwie posortowane tablice:

```
char *strings[] = { "Alex", "Bill","Bill","Celine", "Dexter", "Forest","Forest","Garcia","Garcia","Garcia", "Pedro","Zorro"};
```

```
double numbers[] = {1.34, 1.34 4.34, 5.55, 5.67, 5.67, 5.67 7.76, 8.1, 8.1, 9.12, 11.23};
```

Program, korzystając z funkcji `bsearch` ma sprawdzić czy podane jako **argumenty wywołania programu** wartości występują w odpowiedniej tablicy, a jeżeli tak, to ile razy.

3. (3) Proszę utworzyć strukturę `Klient`, zawierającą pola: `f_name` i `l_name` do przechowywania imienia i nazwiska, pole `code` do przechowywania kodu pocztowego w rzeczywistym formacie (np. 30-111) oraz pole `age`.
Następnie,proszę utworzyć 6-cio elementową tablicę struktur `Klient` i wypełnić danymi wczytanymi z pliku tekstowego .
Tablicę proszę posortować korzystając z funkcji `qsort` według **nazwiska** (pierwszy klucz sortowania), a następnie według **imienia** (drugi klucz sortowania) i wypisać.

```
Wynik:
      Abacka Anna kod 30-143 wiek 22
      Abacka Ewa kod 30-203 wiek 15
      Abacki Jan kod 30-103 wiek 12
      Dadacki Tomasz kod 30-153 wiek 42
      Edacki Jan kod 34-104 wiek 14
      Zazacki Janek kod 30-203 wiek 12
```

4. W programie proszę utworzyć strukturę `struct wektor` do przechowywania współrzędnych wektora 3D, oraz strukturę `abc`, która ma dwa pola: `vect` typu `struct wektor` oraz `len` typu `double`, gdzie zapisana jest długość wektora z pola `vect`.
Proszę zaalokować dynamiczną tablicę struktur `abc` o rozmiarze podanym jako **parametr wywołania programu**, wypełnić liczbami losowymi pola `vect` w elementach tablicy.

- (2) Proszę napisać funkcje `wekt_len` oraz `f_d` zgodnie z podanymi prototypami
  `double wekt_len (struct wector );` - funkcja licząca długość wektora - pole `len`
  `f_d (struct abc *, int );` - procedura wypełniająca w tablicy `abc` pole `len` z wykorzystaniem funkcji `wekt_len`
- Wypisać tablicę struktur `abc` w formacie
  `numer struktury wsp_x wsp_y wsp_z dlugosc`
- (2) Proszę posortować z użyciem funkcji `qsort` tablicę struktur rosnąco względem długości wektora
- Wypisać posortowaą tablicę struktur `abc` w formacie
  `numer struktury wsp_x wsp_y wsp_z dlugosc`

Status przesłanego zadania

Status przesłanego zadania	Przesłane do oceny
Stan oceniania	Nieocenione
Termin oddania	poniedziałek, 27 kwietnia 2020, 14:25

Pozostały czas	Zadanie zostało złożone 1 sek przed terminem
Ostatnio modyfikowane	poniedziałek, 27 kwietnia 2020, 14:24
Przesyłane pliki	zad1.c 27 kwietnia 2020, 14:24 zad2.c 27 kwietnia 2020, 14:24 zad3.c 27 kwietnia 2020, 14:24 zad4.c 27 kwietnia 2020, 14:24
Komentarz do przesłanego zadania	Komentarze (1)

◄ qsort i bsearch

Przejdź do...

IS_L9 ►

Platforma e-Learningowa obsługiwana jest przez:
Centrum e-Learningu AGH oraz Centrum Rozwiązań Informatycznych AGH

Pobierz aplikację mobilną