**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 5\_02 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | Symulacja centrali telefonicznej | | |

1. Treść

Zaimplementuj algorytm realizujący symulację rozmów telefonicznych przez centralę telefoniczną przy założeniach:

- 3 dostępne linie telefoniczne

- 3 rodzaje klientów

- P (PLATINIUM)

- G (GOLD)

- S (SILVER)

Metoda realizacji

Utworzenie trzech kolejek po każdej dla innego priorytety klienta,

* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe *(opis)*

Kod klienta– kod– wczytywana z pliku

Numer klienta-numer-wczytywany z pliku

Czas rozmowy -czas-wczytywany z pliku

* + 1. Dane wyjściowe *(opis)*

Informacje o połączonych klientach dla każdej linii od pierwszego do ostatniego klienta – wyświetlone na ekranie i w pliku .txt

1. Realizacja
   1. Algorytm *(lista kroków lub schemat blokowy)*

1.Deklaracja struktury klienta

2.Deklaracja struktury linii telefonicznej

3.Utworzenie wskaźnikowo trzech kolejek po każdej dla odpowiedniego priorytetu

4.Uzupełnienie pierwszej kolejki klientami PLATINIUM, w kolejności wejścia na linie

5.Uzupełnienie pierwszej kolejki klientami GOLD, w kolejności wejścia na linie

6.Uzupełnienie pierwszej kolejki klientami SILVER, w kolejności wejścia na linie

7.Utworzenie wskaźnikowo kolejki przechowywującej wszystkich klientów, na czele klienci PLATINIUM, za nimi klienci GOLD i na końcu klienci SILVER.

8.Utworzenie wskaźnikowo trzech kolejek po każdej dla linii telefonicznej

9.Wpuszczenie trzech pierwszych klientów na wolne infolinie

10.Sprawdzenie który z klientów ma najmniejszy czas rozmowy

11Odjęcie tego czasu od czasu klientów na innych liniach i wpuszczenie na linie z klientem o najniższym pozostałym czasie rozmowy nowego klienta

12Jesli kolejka klientów nie jest pusta wróć do kroku 10.

13.Wypisywanie na ekranie i w pliku .txt linii telefonicznych i klientów, którzy z nich skorzystali w kolejności od pierwszego do ostatniego

* 1. Kod źródłowy

//================================================

//Zadanie 5\_02 jezyk C++

//Symulacja laczenia rozmow telefonicznych

//WCY21IY1S1 Borkowski Kamil

//================================================

#include <iostream>

#include <cstdlib>

#include <fstream>

using namespace std;

struct Klient

{

char kod\_klienta;

int numer\_klienta;

int czas\_rozmowy;

Klient\* nastepny;

};

struct Infolinia1

{

int aktualny\_rozmowca;

int pozostaly\_czas;

Infolinia1\* nastepny\_rozmowca;

};

struct Infolinia2

{

int aktualny\_rozmowca;

int pozostaly\_czas;

Infolinia2\* nastepny\_rozmowca;

};

struct Infolinia3

{

int aktualny\_rozmowca;

int pozostaly\_czas;

Infolinia3\* nastepny\_rozmowca;

};

int main()

{

ifstream we;

we.open("we.txt");

ofstream wy;

wy.open("wy.txt");

//deklaracja kolejek dla klientow o rozych priorytetach

Klient\* Phead = NULL;

Klient\* Ghead = NULL;

Klient\* Shead = NULL;

Klient\* Pcurr = NULL;

Klient\* Gcurr = NULL;

Klient\* Scurr = NULL;

//utworzenie kolejek

char kod;

int numer;

int czas;

while (we>>kod>>numer>>czas)

{

if(kod=='P')//wypelnianie kolejki PLATINIUM

{

Klient\* nowy = (Klient\*)malloc(sizeof(Klient));

nowy->kod\_klienta = kod;

nowy->numer\_klienta = numer;

nowy->czas\_rozmowy = czas;

nowy->nastepny = NULL;

if(Phead == NULL)

{

Phead = nowy;

Pcurr = nowy;

}

else

{

Pcurr->nastepny = nowy;

Pcurr = nowy;

}

}

if(kod=='G')//wypelnianie kolejki GOLD

{

Klient\* nowy = (Klient\*)malloc(sizeof(Klient));

nowy->kod\_klienta = kod;

nowy->numer\_klienta = numer;

nowy->czas\_rozmowy = czas;

nowy->nastepny = NULL;

if(Ghead == NULL)

{

Ghead = nowy;

Gcurr = nowy;

}

else

{

Gcurr->nastepny = nowy;

Gcurr = nowy;

}

}

if(kod=='S')//wypelnianie kolejki SILVER

{

Klient\* nowy = (Klient\*)malloc(sizeof(Klient));

nowy->kod\_klienta = kod;

nowy->numer\_klienta = numer;

nowy->czas\_rozmowy = czas;

nowy->nastepny = NULL;

if(Shead == NULL)

{

Shead = nowy;

Scurr = nowy;

}

else

{

Scurr->nastepny = nowy;

Scurr = nowy;

}

}

}

//Polaczenie kolejek w jedna, z klientami P na czele i S na koncu

Klient\* head = NULL;

Klient\* curr = NULL;

Pcurr = Phead;//dodanie kolejki P

while(Pcurr != NULL)

{

Klient\* nowy = (Klient\*)malloc(sizeof(Klient));

nowy->kod\_klienta = Pcurr->kod\_klienta;

nowy->numer\_klienta = Pcurr->numer\_klienta;

nowy->czas\_rozmowy = Pcurr->czas\_rozmowy;

nowy->nastepny = NULL;

if(head == NULL)

{

head = nowy;

curr = nowy;

}

else

{

curr->nastepny = nowy;

curr = nowy;

}

Pcurr = Pcurr->nastepny;

}

Gcurr = Ghead;//dodanie kolejki G

while(Gcurr != NULL)

{

Klient\* nowy = (Klient\*)malloc(sizeof(Klient));

nowy->kod\_klienta = Gcurr->kod\_klienta;

nowy->numer\_klienta = Gcurr->numer\_klienta;

nowy->czas\_rozmowy = Gcurr->czas\_rozmowy;

nowy->nastepny = NULL;

if(head == NULL)

{

head = nowy;

curr = nowy;

}

else

{

curr->nastepny = nowy;

curr = nowy;

}

Gcurr = Gcurr->nastepny;

}

Scurr = Shead;//dodanie kolejki S

while(Scurr != NULL)

{

Klient\* nowy = (Klient\*)malloc(sizeof(Klient));

nowy->kod\_klienta = Scurr->kod\_klienta;

nowy->numer\_klienta = Scurr->numer\_klienta;

nowy->czas\_rozmowy = Scurr->czas\_rozmowy;

nowy->nastepny = NULL;

if(head == NULL)

{

head = nowy;

curr = nowy;

}

else

{

curr->nastepny = nowy;

curr = nowy;

}

Scurr = Scurr->nastepny;

}

//deklaracja linitelefonicznych

Infolinia1\* head1 = NULL;

Infolinia2\* head2 = NULL;

Infolinia3\* head3 = NULL;

Infolinia1\* curr1 = NULL;

Infolinia2\* curr2 = NULL;

Infolinia3\* curr3 = NULL;

//Polaczenie pierwszych trzech klientow do kazdej z lini

curr = head;

//pierwszy klient pierwszej infolini

Infolinia1\* nowy = (Infolinia1\*)malloc(sizeof(Infolinia1));

nowy->aktualny\_rozmowca = curr->numer\_klienta;

nowy->pozostaly\_czas = curr->czas\_rozmowy;

nowy->nastepny\_rozmowca = NULL;

head1 = nowy;

curr1 = nowy;

curr = curr->nastepny;

//pierwszy klient drugiej infolini

Infolinia2\* nowy2 = (Infolinia2\*)malloc(sizeof(Infolinia2));

nowy2->aktualny\_rozmowca = curr->numer\_klienta;

nowy2->pozostaly\_czas = curr->czas\_rozmowy;

nowy2->nastepny\_rozmowca = NULL;

head2 = nowy2;

curr2 = nowy2;

curr = curr->nastepny;

//pierwszy klient trzeciej infolini

Infolinia3\* nowy3 = (Infolinia3\*)malloc(sizeof(Infolinia3));

nowy3->aktualny\_rozmowca = curr->numer\_klienta;

nowy3->pozostaly\_czas = curr->czas\_rozmowy;

nowy3->nastepny\_rozmowca = NULL;

head3 = nowy3;

curr3 = nowy3;

curr = curr->nastepny;

int min\_czas;

while(curr != NULL)

{

//sprawdzenie kto pierwszy zejdzie z linii

if( (curr1->pozostaly\_czas <= curr2->pozostaly\_czas) && (curr1->pozostaly\_czas <= curr3->pozostaly\_czas) ) min\_czas=curr1->pozostaly\_czas;

if( (curr2->pozostaly\_czas<=curr1->pozostaly\_czas) && (curr2->pozostaly\_czas<=curr3->pozostaly\_czas) ) min\_czas=curr2->pozostaly\_czas;

if( (curr3->pozostaly\_czas<=curr1->pozostaly\_czas) && (curr3->pozostaly\_czas<=curr2->pozostaly\_czas) ) min\_czas=curr3->pozostaly\_czas;

//wejscie na linie kolejnego klienta

if(curr1->pozostaly\_czas==min\_czas)

{

Infolinia1\* nowy3 = (Infolinia1\*)malloc(sizeof(Infolinia1));

nowy3->aktualny\_rozmowca = curr->numer\_klienta;

nowy3->pozostaly\_czas = curr->czas\_rozmowy;

nowy3->nastepny\_rozmowca = NULL;

curr1->nastepny\_rozmowca = nowy3;

curr1 = nowy3;

curr3->pozostaly\_czas = curr3->pozostaly\_czas - min\_czas;

curr2->pozostaly\_czas = curr2->pozostaly\_czas - min\_czas;

}

if(curr3->pozostaly\_czas==min\_czas)

{

Infolinia3\* nowy3 = (Infolinia3\*)malloc(sizeof(Infolinia3));

nowy3->aktualny\_rozmowca = curr->numer\_klienta;

nowy3->pozostaly\_czas = curr->czas\_rozmowy;

nowy3->nastepny\_rozmowca = NULL;

curr3->nastepny\_rozmowca = nowy3;

curr3 = nowy3;

curr1->pozostaly\_czas = curr1->pozostaly\_czas - min\_czas;

curr2->pozostaly\_czas = curr2->pozostaly\_czas - min\_czas;

}

if(curr2->pozostaly\_czas==min\_czas)

{

Infolinia2\* nowy3 = (Infolinia2\*)malloc(sizeof(Infolinia2));

nowy3->aktualny\_rozmowca = curr->numer\_klienta;

nowy3->pozostaly\_czas = curr->czas\_rozmowy;

nowy3->nastepny\_rozmowca = NULL;

curr2->nastepny\_rozmowca = nowy3;

curr2 = nowy3;

curr3->pozostaly\_czas = curr3->pozostaly\_czas - min\_czas;

curr1->pozostaly\_czas = curr1->pozostaly\_czas - min\_czas;

}

curr = curr->nastepny;

}

cout<<"Uzytkownicy na linii 1"<<endl;

wy<<"Uzytkownicy na linii 1"<<endl;

curr1 = head1;

while(curr1 != NULL)

{

cout <<curr1->aktualny\_rozmowca<<endl;

wy <<curr1->aktualny\_rozmowca<<endl;

curr1 = curr1->nastepny\_rozmowca;

}

cout<<endl;

wy<<endl;

cout<<"Uzytkownicy na linii 2"<<endl;

wy<<"Uzytkownicy na linii 2"<<endl;

curr2 = head2;

while(curr2 != NULL)

{

cout <<curr2->aktualny\_rozmowca<<endl;

wy <<curr2->aktualny\_rozmowca<<endl;

curr2 = curr2->nastepny\_rozmowca;

}

cout<<endl;

wy<<endl;

cout<<"Uzytkownicy na linii 3"<<endl;

wy<<"Uzytkownicy na linii 3"<<endl;

curr3 = head3;

while(curr3 != NULL)

{

cout <<curr3->aktualny\_rozmowca<<endl;

wy <<curr3->aktualny\_rozmowca<<endl;

curr3 = curr3->nastepny\_rozmowca;

}

cout<<endl;

wy<<endl;

//zwalnianie pamieci

Pcurr= Phead;

while(Pcurr != NULL)

{

Klient\* pom = Pcurr;

Pcurr = Pcurr->nastepny;

free(pom);//zwolniamy pammiec

}

Gcurr= Ghead;

while(Gcurr != NULL)

{

Klient\* pom = Gcurr;

Gcurr = Gcurr->nastepny;

free(pom);//zwolniamy pammiec

}

Scurr= Shead;

while(Scurr != NULL)

{

Klient\* pom = Scurr;

Scurr = Scurr->nastepny;

free(pom);//zwolniamy pammiec

}

curr= head;

while(curr != NULL)

{

Klient\* pom = curr;

curr = curr->nastepny;

free(pom);//zwolniamy pammiec

}

curr1= head1;

while(curr1 != NULL)

{

Infolinia1\* pom = curr1;

curr1 = curr1->nastepny\_rozmowca;

free(pom);//zwolniamy pammiec

}

curr2= head2;

while(curr2 != NULL)

{

Infolinia2\* pom = curr2;

curr2 = curr2->nastepny\_rozmowca;

free(pom);//zwolniamy pammiec

}

curr3= head3;

while(curr3 != NULL)

{

Infolinia3\* pom = curr3;

curr3 = curr3->nastepny\_rozmowca;

free(pom);//zwolniamy pammiec

}

we.close();

wy.close();

return 0;

}

* 1. Dane wejściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

* 1. Dane wyjściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

1. Wnioski *(złożoność asymptotyczna algorytmu przy użyciu notacji* ***O lub innej i inne wnioski)***

Złożoność obliczeniowa programu wynosi: O(n), zależna od ilości klientów .