Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów

Faktury

autor	Maciej Sowiński
prowadzący	Mgr inż. Grzegorz Wojciech Kwiatkowski
rok akademicki	2019/2020
kierunek	informatyka
rodzaj studiów	SSI
semestr	2
termin laboratorium	środa, 13:45 – 15:15
sekcja	62
termin oddania sprawozdania	2020-09-18

1 Treść zadania

Program do zarządzania fakturami. Na wejściu wczytujemy plik z fakturami (unikalny numer faktury, data wystawienia i terminu płatności , dane osobowe sprzedawcy i nabywcy, zapłacona kwota oraz zestawienie towarów/usług (nazwa, ilość, cena netto, podatek)). Program po uruchomieniu wyświetla menu opcji dzięki którym możemy:

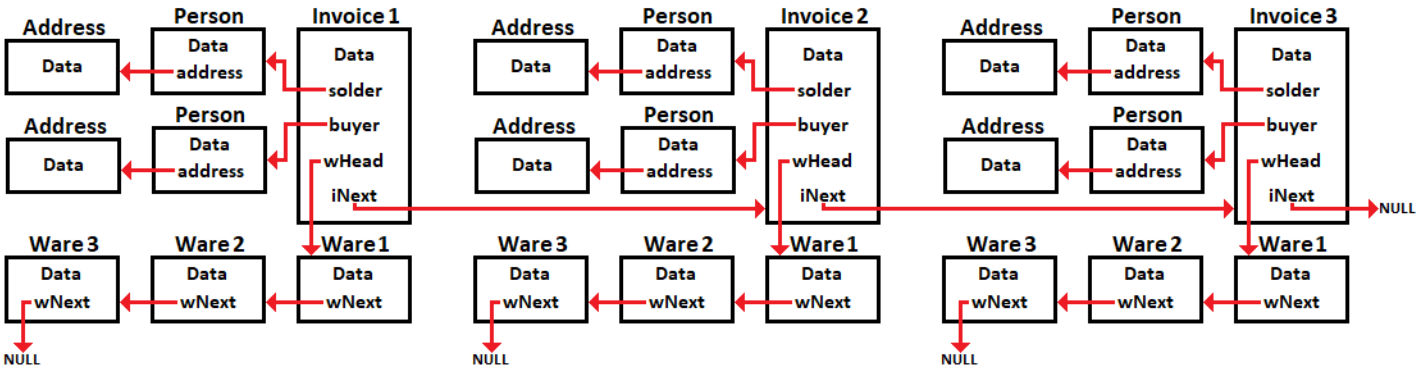
- wyświetlać listę faktur
- wystawiać fakturę
- wprowadzić korektę faktury
- usunąć fakturę
- wyszukiwać faktury po dacie
- wyszukiwać faktury nieopłacone
- zapisać faktury do nowego lub już istniejącego pliku

2 Analiza zadania

Zagadnie przedstawia problem zarządzania fakturami. Dane w pliku który zawiera wiele faktur muszą być w odpowiednim formacie, a miedzy fakturami powinien znaleźć się ustalony i jednakowy separator, tak samo we wnętrzu faktur ustalonymi separatorami musi być oddzielona lista towarów/usług.

2.1 Struktury danych

W programie wykorzystano dwie listy jednokierunkowe. Lista przechowuje dane w swoich elementach. Element oprócz potrzebnych w zadaniu danych posiada wskaźnik na następny element. Lista faktur dla każdej faktury posiada oddzielną listę towarów/usług. Dodatkowo zastosowano struktury Person (przestawia osobę) i Address (przedstawia adres). Powyższe struktury danych powodują ich logiczne i intuicyjne ułożenie co umożliwia wygodne korzystanie z treści w nich zawartych. Program oferuje możliwość wystawiania, korekty i usuwania wielu elementów dynamicznie, szukania oraz wyświetlania wszystkich faktur lub tylko tych spełniających dane kryteria.



Rysunek 1 wygląd użytych w programie struktur

2.2 Algorytmy

Program wykonuje dodawanie elementu odczytanego z pliku do listy umieszczając go na końcu. W przypadku usuwania elementu wybranego przez użytkownika z listy (o ile istnieje), robi to w dowolnym jego ustawieniu na liście (początku, końcu, środku). Wyszukuje elementy spełniające parametr (data, stan zapłaty) podany przez użytkownika. Wyświetla jeden lub wiele elementów listy, a na konkretnym elemencie może dokonać korekty lub usunięcia w zależności od wybranej opcji.
Generuje unikalny numer faktury na podstawie aktualnej daty i pozostałych faktur znajdujących się na liście. Porównuje elementy listy takie jak numer faktury , data i kwota zapłaty.
Wszystkie powyższe funkcje operują na listach i przechodzą przez nie w sposób iteracyjny.

3 Specyfikacja zewnętrzna

Program operuje na plikach tekstowych. Nazwa pliku wejściowego i opcjonalnego wyjściowego jest zawarta w kodzie programu ze względów bezpieczeństwa. Jeśli plik znajduje się w folderze programu to ścieżka (**PATH**) jest pusta.

```
14      const char *PATH = "";                  //ŚCIEŻKA DO PLIKU
15      char fileName[100] = "invoice.txt";     //NAZWA PLIKU WEJŚCIOWEGO
```

Zdjęcie 1 ścieżka do i nazwa pliku wejściowego

```
63    strcpy(fileName, src: "new_invoice.txt");    //NAZWA NOWEGO PLIKU WYJŚCIOWEGO
```

Zdjęcie 2 nazwa opcjonalnego pliku wyjściowego

Jeżeli odczytanie pliku zakończyło się pomyślnie to program zwróci komunikat:

```
File read done
10 invoices loaded
```

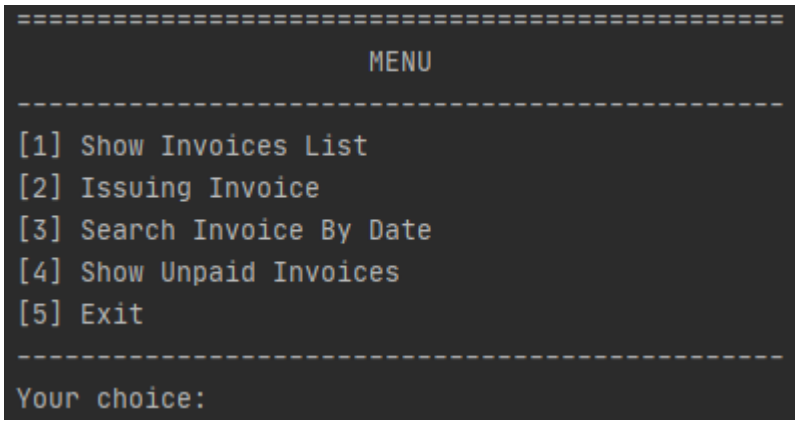
Zdjęcie 3 komunikat poprawnego odczytania pliku wraz z ilością odczytanych faktur

W przeciwnym wypadku komunikat będzie następujący:

```
file invoices.txt not found
```

Zdjęcie 4 komunikat błędnego odczytania pliku lub jego braku

Program po uruchomieniu wyświetla główne menu opcji.



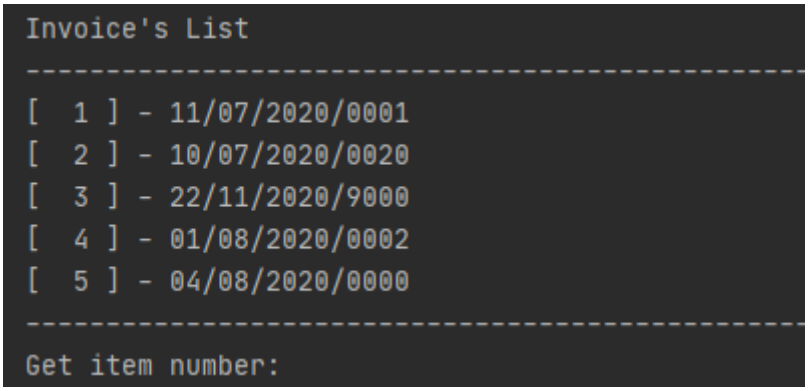
Zdjęcie 5 wygląd głównego menu opcji

Opcje mogą być wybierane w dowolnej kolejności i wielokrotnie po przez wybranie odpowiedniej cyfry znajdującej się obok polecenia. W przypadku wpisania liczby z poza wyświetlonego zakresu, lub w formacie nie zgodnym z tym o którym jest mowa w pytaniu na konsolę zostanie wyświetlony stosowny komunikat.

Zdjęcie 6 komunikat o błędnych danych

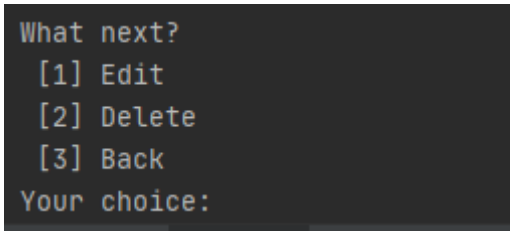
3.1 Show Invoice List

Wybierając opcję **`Show Invoice List`** zostanie wyświetlona lista numerów faktur odczytanych z pliku wraz z numerem na liście im przyporządkowanym.



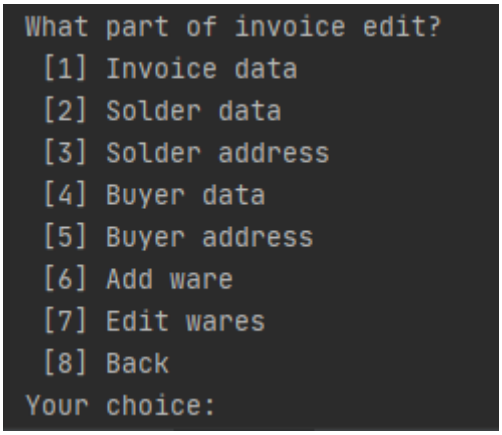
Zdjęcie 7 wygląd listy faktur

Po wybraniu odpowiedniej faktury zostanie ona wyświetlona, a pod nią pojawi się menu działań dla wybranej faktury.



Zdjęcie 8 wygląd menu dla wybranej faktury

Opcja **`Delete`** usunie fakturę z listy, opcja **`Back`** powróci do głownego menu opcji. Natomiast opcja **`Edit`** wyświetli listę rzeczy możliwych do zedytowania.

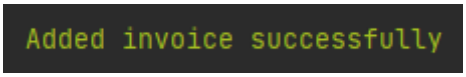


Zdjęcie 9 wygląd menu edycji dla wybranej faktury

Po wybraniu numeru odpowiadającego części faktury, będzie można dokonać jej korekcji wpisując nowe dane. Opcja **`Edit wares`** powoduje wyświetlenie listy nazw towarów/usług, a następne kroki postępowania są analogiczne jak w przypadku edycji listy faktur. Opcja **`Back`** wraca do poprzedniego menu.

3.2 Issuing Invoice

Opcja ta przeznaczona jest do wystawiania nowej faktury. Zadawane są kolejno pytania o dane dotyczące faktury, podliczana jest suma za towary/usługi, generowany jest unikalny numer faktury i wstawiana aktualna data. Na końcu wyświetlane są odpowiedni komunikat i nowo dodana faktura.



Zdjęcie 10 komunikat po poprawnym dodaniu faktury

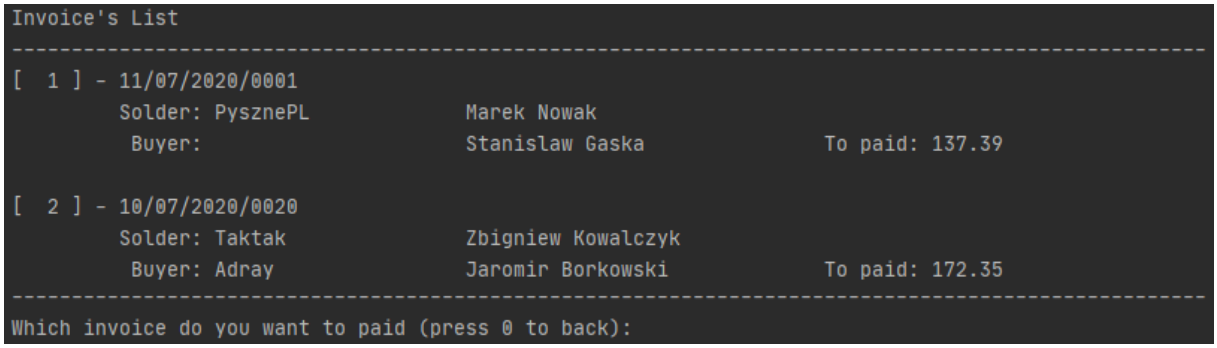
Po wystawieniu faktury program przekierowuje do głównego menu.

3.3 Search Invoice By Date

Opcja ta prosi o podanie daty w formacie (dd.mm.rrrr). Następnie wyświetla listę faktur wystawionych w tym dniu i pozwala wykonać na nich takie same operacje jak w punkcie 3.1.

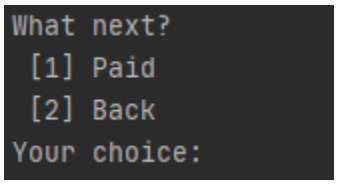
3.4 Show Unpaid Invoices

Opcja wyświetla listę nieopłaconych faktur.



Zdjęcie 11 wygląd nieopłaconej listy faktur

Po wybraniu faktury (można też się wycofać do menu klikając ”0”), wyświetlają się dane do przelewu oraz menu płatności.

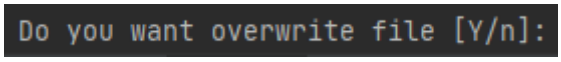


Zdjęcie 12 wygląd menu płatności

Opcja **`Paid`** zmieni fakture na opłaconą, opcja **`Back`** powróci do głownego menu opcji.

3.5 Exit

Opcja pyta czy nadpisać obecny plik faktur czy utworzyć nowy, następnie zapisuje dane do wybranego pliku i kończy program.



Zdjęcie 13 pytanie o nazwę pliku

4 Specyfikacja wewnętrzna

W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji (operacji wykonywanych na fakturach i odczytu/zapisu pliku).

4.1 Operacje wejścia/wyjścia

Pliki ”input” zawierają funkcje wejścia, przykładami są:

```
39      while (fgets(string, n: 50, (FILE *) ptr) != NULL) {...}
```

Zdjęcie 14 pętla odczytująca słowa z pliku (fragment funkcji **`readDataFromFile`**)

```
150             letter = (char) getchar();
```

Zdjęcie 15 odczytanie znaku i przypisanie go do zmiennej (fragment funkcji **`readLine`**)

Pliki ”output” zawierają funkcje wyjścia, przykładami są:

```
70      fprintf(fptr, format: "%s %s\n%s %s\n",
71              address->street,
72              address->homeNumber,
73              address->postalCode,
74              address->city
75      );
```

Zdjęcie 16 zapisanie w pliku danych adresowych (fragment funkcji **`filePrintAddress`**)

```
117     printf( format: "\nWhat next?\n "
118             "[1] Edit\n "
119             "[2] Delete\n "
120             "[3] Back\n"
121             "Your choice:");
```

Zdjęcie 17 wyświetlenie na konsolę menu opcji (fragment funkcji **`printOptions`**)

4.2 Dynamiczna alokacja pamięci

Zmienne dynamiczne są tworzone i usuwane w takcie działania programu.

```
14      struct Invoice *invoice;
15      invoice = (struct Invoice *) malloc(sizeof(struct Invoice));
```

Zdjęcie 18 alokacja pamieci z użyciem **`malloc`** (fragment funkcji **`createInvoice`**)

```
217     while (invoice->wHead) {
218         struct Ware *tmp = invoice->wHead->wNext;
219         free(invoice->wHead);
220         invoice->wHead = tmp;
221     }
222     free(invoice);
```

Zdjęcie 19 zwalnianie pamięci z użyciem **`free`** (fragment funkcji **`deleteInvoice`**)

4.3 Obsługa plików

Odczytem i zapisem danych pliku zajmują się kolejno funkcje **`readDataFromFile`** i **`saveDataToFile.`**

```
16      FILE *ptr;
22      if ((ptr = fopen(readPath, modes: "r")) == NULL)
39      while (fgets(string, n: 50, (FILE *) ptr) != NULL) {...}
138     fclose(ptr);
```

Zdjęcie 20 elementy obsługi i działania na pliku (fragmenty funkcji **`readDataFromFile`**)

4.4 Obsługa łańcuchów znakowych

Łancuchy znaków w C przechowywane są w pamięci jako następujący po sobie ciąg znaków (char), a zmienna przechowująca łańcuch znaków to tak naprawdę wskaźnik do pamięci.

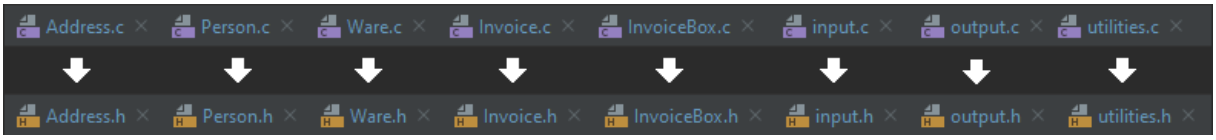
```
73      memset(string, c: '\0', n: 150);
74
75      strcat(string, a);
76      strcat(string, src: " ");
77      strcat(string, b);
```

Zdjęcie 21 obsługa C-string (fragment funkcji **`concatenationStrings`**)

4.5 Podział na pliki źródłowe i nagłówkowe

W programie pliki są podzielone tematycznie, a funkcje w nich zawarte wykonują operacje w zakresie danego tematu.

Dodatkowo pliki są pogrupowane w tematycznych folderach dla zwiększenia przejrzystości. Pliki źródłowe mają swoje odpowiedniki w plikach nagłówkowych.



Zdjęcie 22 podział na pliki zródłowe i nagłówkowe

4.6 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5 Testowanie

Program został przetestowany na kilku plikach. Brak pliku powoduje zgłoszenie błędu. Plik pusty nie powoduje zgłoszenia błędu, ale utworzenie pustego pliku wynikowego. Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Dzięki dobremu rozplanowaniu struktur zadanie nie było bardzo skomplikowane jednak czasochłonne. Odpowiednia ilość tematycznie utworzonych plików .c i .h znacznie ułatwiła dopisywanie nowych funkcji i korzystanie z już istniejących, a także szukania najprostszego rozwiązania problemów stawianych przez kolejne podpunkty zadania. Program jest otwarty na dalszy rozwój o dodatkowe struktury czy funkcje.

Projekt pomógł zrozumieć jak działa lista oraz utrwalił posługiwanie się wskaźnikami. Największym problemem był brak niektórych właściwości(w szczególności operujących na łańcuchach znaków) i ułatwień, które posiada język C++ oraz wiele współczesnych języków.