Laboratorium 4

Przekazywanie parametrów do funkcji przez wartość, przez wskaźnik (adres), przez referencję (zmienną) oraz operacje na łańcuchach znaków

Proszę wykonać obowiązkowo zadanie 1a, 1b, 2, 3. Osoby ambitne mogą wykonać również zadanie 4.

Zadanie 1a oraz 1b proszę zrobić jako oddzielne programy (każdy w innym pliku źródłowym).

Zadania 2, 3 oraz 4 (nieobowiązkowo) proszę zrobić jako jeden program, w którym każde zadanie jest oddzielną funkcją wywoływaną z prostego menu w funkcji *main().*

UWAGA: Każdy program musi być w komentarzu podpisany ! Po uruchomieniu każdy program powinien na początku wyświetlać na ekranie imię i nazwisko autora. Do oceny proszę wysłać tylko plik źródłowy tzn. plik z rozszerzeniem *.cpp.

Program przykładowy nr 1

Proszę przeanalizować program przykładowy *row_kwa.cpp*, który ilustruje sposób podziału programu na kilka funkcji. Program oblicza pierwiastki równania kwadratowego.

Funkcje `CzytajDane` oraz `WypiszPierwiastki` zapewniają komunikację z użytkownikiem programu tzn. wczytują współczynniki równania, wypisują pierwiastki i inne komunikaty. Sposób implementacji tych funkcji zależy od wykorzystywanej biblioteki wejścia-wyjścia. W tym programie została wykorzystana biblioteka `stdio.h`, a do wypisywania i wczytywania danych zostały użyte funkcje `printf` oraz `scanf`.

Funkcje `Delta` oraz `ObliczPierwiastki` wykonują wszystkie obliczenia. Te funkcje nie wypisują żadnych komunikatów na ekranie, a więc są niezależne od wykorzystywanej biblioteki wejścia-wyjścia.

Proszę zwrócić szczególną uwagę na dobór sposobu przekazywania parametrów w poszczególnych funkcjach.

Zadanie 1a

Proszę napisać program który rozwiązuje układ dwóch równań liniowych :

$$\begin{cases} a1*x+b1*y=c1 \\ a2*x+b*2y=c2 \end{cases}$$

Proszę zastosować metodę wyznaczników (Wzory Cramera):

$$W=\begin{vmatrix} a1 & b1 \\ a2 & b2 \end{vmatrix}=a1*b2-a2*b1$$

$$Wx = \begin{vmatrix} c1 & b1 \\ c2 & b2 \end{vmatrix} = c1 * b2 - c2 * b1$$

$$Wy = \begin{vmatrix} a1 & c1 \\ a2 & c2 \end{vmatrix} = a1 * c2 - a2 * c1$$

Możliwe są trzy przypadki:

- Jeżeli $W \neq 0$ to układ posiada dokładnie 1 jedno rozwiązanie:

$$x = \frac{Wx}{W} \qquad y = \frac{Wy}{W}$$

- Jeżeli $W = 0 \quad Wx = 0 \quad Wy = 0$ to układ posiada nieskończenie wiele rozwiązań.
- Jeżeli $W = 0 \quad Wx \neq 0 \quad Wy \neq 0$ to układ jest sprzeczny tzn. nie posiada żadnego rozwiązania.

Proszę podzielić program na funkcje realizujące komunikację z użytkownikiem oraz funkcje wykonujące obliczenia. Proponuję następujące funkcje:

```
void  CzytajRownanie(float &a, float &b, flaat &c);
```

- funkcja wczytuje współczynniki jednego równania. $a * x + b * y = c$

```
void WypiszRozwiązanie( int N, float x, float y);
```

- funkcja wypisuje rozwiązanie układu równań lub inny komunikat

```
float ObliczWyznacznik(float p1, float p2, float p3,float p4);
```

-funkcja oblicza wyznacznik $\begin{vmatrix} p1 & p2 \\ p3 & p4 \end{vmatrix} = p1 * p4 - p3 * p2$

```
int ObliczRozwiązanie(float a1, float b1, float c1,
                      float a2, float b2, float c2,
                      float &x, float &y);
```

- funkcja oblicza rozwiązanie układu równań i zwraca:
  - 2 – nieskończenie wiele rozwiązań
  - 1 – jest jedno rozwiązanie
  - 0 – brak rozwiązań (układ sprzeczny)

Funkcja `main` może mieć wówczas następującą postać:

```
Int main(int argc, char*argv[] )
{ float A1, B1, C1;  // pierwsze równanie,
  float A2, B2, C2;  // drugie równanie,
  float X, Y;        // rozwiązanie,
  int N;             // liczba rozwiazan.

  CzytajRownanie(A1, B1, C1);
  CzytajRownanie(A2, B2, C2);
  N = ObliczRozwiązanie(A1, B1, C1, A2, B2, C2, X, Y);
  WypiszRozwiązanie( N, X, Y);
  getch();
  return 0;
}
```

Zadanie 1b

Proszę zmodyfikować program z poprzedniego zadania tak, by parametry przekazywane do funkcji przez referencję zastąpić parametrami przekazywanymi przez wskaźnik.

Uwaga: Zmodyfikowany program proszę umieścić nowym pliku źródłowym.

Zadanie 2

Napisz funkcję, która z podanego w parametrze łańcucha znaków usuwa wszystkie cyfry (znaki od '0' do '9'). Funkcja powinna zwracać jako wynik swojego działania liczbę znalezionych i usuniętych cyfr. Zmodyfikowany łańcuch znaków powinien być zapamiętany pod tym samym adresem ca łańcuch pierwotny.

Zadanie 3 (kontynuacja zadania 2)

Napisz funkcję, która z podanego w parametrze łańcucha znaków usunie komentarze. Jako komentarz należy traktować wszystkie znaki rozpoczynające począwszy od sekwencji "/*" do sekwencji "*/" oraz wszystkie znaki począwszy od ciągu "//" do końca łańcucha. Zmodyfikowany łańcuch znaków powinien być zapamiętany pod tym samym adresem ca łańcuch pierwotny. Funkcja powinna zwracać adres początku łańcucha wynikowego.

Zadanie 4 (nieobowiązkowe – dla ambitnych)

Proszę zadeklarować kilka zmiennych globalnych i lokalnych różnych typów (np. `char`, `int`, `long`, `float`, `double`) i przypisać im różne wartości początkowe. Następnie napisz instrukcje wykonujące następujące operacje:

- Wypisz na ekranie adresy oraz wartości wszystkich zadeklarowanych zmiennych globalnych i lokalnych, (porównaj czym różnią się adresy zmiennych globalnych i lokalnych.
- Narysuj na kartce papieru „bajtową mapę pamięci", na której zaznacz adresy i obszary zajmowane przez poszczególne zmienne. (ilość pamięci zajmowanej przez zmienne określonych typów były podane w materiałach do pierwszego wykładu)
- Napisz instrukcje, które wyświetlą w kodzie szesnastkowym (heksadecymalnie) zawartości poszczególnych bajtów kodujących poszczególne zmienne. **Uwaga:** Dla każdej zmiennej należy wypisać tyle bajtów ile ta zmienna zajmuje miejsca w pamięci np. dla zmiennej `A` typu `int` wydruk powinien wyglądać następująco:
  ```
  Zmienna A: Adres:0x45674523  Wartosc:10    Bajty:0x00 0x00 0x00 0x0A
  ```
- Napisz instrukcje zapisu nowej wartości każdej zmiennej za pomocą adresu tej zmiennej (tzn. dla każdej zmiennej utwórz wskaźnik, który ma przypisany adres tej zmiennej i wskaż miejsce zapisu do pamięci za pomocą tego wskaźnika – wykorzystaj operator wyłuskania), a następnie wydrukuj ponownie wartości wszystkich zadeklarowanych zmiennych oraz ich reprezentację w poszczególnych bajtach.
- Napisz instrukcje zapisu wartości zmiennej za pomocą adresu innej zmiennej (wskaż miejsce zapisu za pomocą wskaźnika zawierającego adres innej zmiennej, do którego jest dodana odpowiednia liczba całkowita), a następnie wydrukuj ponownie wartości wszystkich zadeklarowanych zmiennych.

Wskazówki do zadania 4

- Zmienne globalne to zmienne zadeklarowane poza ciałem jakiejkolwiek funkcji. Zwykle zmienne globalne są deklarowane na początku programu przed deklaracjami wszystkich funkcji. Do zmiennych globalnych można odwoływać się we wszystkich funkcjach, które są zadeklarowane w programie później. Zmiana wartości zmiennej globalnej w jednej funkcji będzie widoczna również w innej funkcji.

- Zmienne lokalne to zmienne zadeklarowane wewnątrz funkcji. Zmiennymi lokalnymi są również parametry funkcji przekazywane przez wartość. Zmienne lokalne zwykle są deklarowane na początku ciała funkcji. Mogą być ona deklarowane wewnątrz pętli `for` lub wewnątrz dowolnego innego bloku instrukcji ograniczonego parą nawiasów klamrowych `{ }`. Zasięg zmiennych lokalnych jest ograniczony wyłącznie do bloku instrukcji `{ }`, w którym zmienne te są zadeklarowane. Do tych zmiennych można się odwoływać tylko w instrukcjach, które są w zasięgu zmiennej tzn. są wewnątrz tego samego bloku instrukcji.

- W różnych funkcjach mogą być deklarowane zmienne lokalne o tych samych nazwach. Zmienne takie są od siebie niezależne tzn. operacje wykonane na zmiennej lokalnej w jednej funkcji nie mają wpływu na wartości zmiennej lokalnej zadeklarowanej w innej funkcji.

- Jeżeli w funkcji jest zadeklarowana zmienna lokalna o nazwie pokrywającej się z nazwą zmiennej globalnej, to wewnątrz tej funkcji odwołania będą zawsze dotyczyły zmiennej lokalnej, a nie globalnej (tzn. zmienna lokalna „przesłania" zmienną globalną.

- Ilość pamięci zajmowanej przez zmienną określonego typu można odczytać za pomocą instrukcji `sizeof(typ zmiennej)`.

- Adres zmiennej (wartość wskaźnika) można wypisać za pomocą instrukcji `printf`, dla której w łańcuchu formatującym jest użyty symbol `"%p"` (zapis hexadecymalny) lub `"%u"`(zapis dziesiętny). np.

```
int x;
int *wsk;

wsk = &x;
printf("Adres zmiennej x = %p", wsk);
printf("Wartość zmiennej x = %d", *wsk);
```