Języki i paradygmaty programowania 1 – studia stacjonarne 2024/25

Lab 9. Funkcje w argumentach funkcji – metoda Newtona. Synonimy nazw typów danych. Struktury. Tablice struktur.

1. **Identyfikator funkcji, wskaźnik do funkcji.**
   - Funkcje nie są zmiennymi;
   - Identyfikator funkcji jest typu „wskaźnik do funkcji” i ma wartość równą adresowi funkcji;
   - Wskaźniki do funkcji można przekazywać jako argumenty aktualne do innych funkcji. Odpowiedni argument formalny musi być wtedy wskaźnikiem do funkcji;
   - Informacja o typie wartości zwracanej przez funkcję znajdującą się na liście argumentów formalnych musi być uwzględniona w deklaracji.
   - `int f();      //f - funkcja zwracająca wartość typu int`
   - `int (*f)();   //f - wskaźnik do funkcji zwracającej wartość typu int`

2. **Napisz funkcję obliczającą wartość :**

$$y = x^2 + f(x)$$

tak by funkcja *f* mogła być dowolną funkcją. W funkcji *main()* policz wartość y dla *f(x)=sin(x)* i *f(x) = cos(x).*

3. **Metoda Newtona.**

Metoda Newtona (zwana metodą stycznych) pozwala na rozwiązanie równania postaci:

$$f(x) = 0$$

Założenia metody w odniesieniu do funkcji f(x):
- Pierwiastek równania α znajduje się w przedziale [a,b].
- Funkcja ma różne znaki na krańcach przedziału, tj. $f(a) * f(b) < 0$.
- Istnieją i są ciągłe f'(x) i f''(x), ponadto f''(x) nie zmienia znaku w [a,b].

Metoda działania:

Niech x_i będzie *i*-tym przybliżeniem pierwiastka, a h_i błędem tego przybliżenia, wtedy

$$\alpha = x_i + h_i$$

po podstawieniu do funkcji otrzymujemy

$$f(x_i + h_i) \equiv 0$$

Rozwijając funkcję w f szereg Taylora z zachowaniem jedynie wyrazu pierwszego rzędu otrzymujemy:

$$0 = f(x_i + h_i) \cong f(x_i) + h_i f'(x_i)$$

stąd otrzymujemy

$$h_i = -\frac{f(x_i)}{f'(x_i)}$$

Ponieważ błąd w *i*-tej iteracji został policzony w sposób przybliżony, zamiast pierwiastka dokładnego α otrzymujemy kolejne przybliżenie pierwiastka

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

dla *i=0,1,2,3…*

Iterację (obliczenia) należy przerwać po skończonej liczbie kroków. Warunek przerwania można zdefiniować jako:

$$|x_{i+1} - x_i| < \varepsilon$$

gdzie ε jest żądaną dokładnością.

Interpretacja geometryczna patrz: https://pl.wikipedia.org/wiki/Metoda_Newtona

Przykład.

Zbuduj program rozwiązujący równanie .

$$sinx - \frac{1}{2}x = 0$$

podając następujące dane: przybliżenie początkowe (punkt startowy), dokładność, maksymalną liczbę iteracji. Obliczanie wartości funkcji dla dowolnego x zapisz w funkcji *ff(),* obliczanie wartości pochodnej zapisz w funkcji *fp()*. Metodę Newtona zapisz w funkcji *newton()*. Funkcja *newton()* ma zwrócić informację o zbieżności procesu iteracyjnego: 0 – jeżeli proces jest zbieżny; 1 – jeżeli proces jest rozbieżny.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double f(double);
double fp(double);
int newton(double *x, int n, double eps);
/*   start 0.5 -> 0     0.8 -> -1.8954  1.0 -> 1.8954 */
int main()
{
	double x0 = 0.8, eps = 0.001;
	int n = 100;

	//  printf("podaj:\n  przybl. dokl. maks.iter.\n");
	//  scanf("%lf %lf %d", &x0,&eps,&n);

	if (!newton(&x0, n, eps))
		printf("rozwiazanie= %lf wart.= %lf\n", x0, f(x0));
	else
		printf("Brak zbieznosci\n");
	system("pause");
}

int newton(double *x, int n, double eps)
{
	double x1;
	int i = 0;
	do
	{
		x1 = -f(*x) / fp(*x);
		*x += x1;
		if (fabs(x1) < eps) return 0;
	} while (i++ < n);
	return 1;
}

double f(double x)
{
	return sin(x) - .5*x;
}

double fp(double x)
{
	return cos(x) - .5;
}
```

4. **Funkcje w argumentach funkcji** – zmodyfikuj program z pkt.2 w taki sposób aby do funkcji `int newton()` przekazać przez argumenty funkcje `double f()` i `double fp()`.
Tak napisany program wykorzystaj do rozwiązania równania:

$$sinx - \frac{1}{2}x = 0$$

oraz

$$tgx - 2x = 0$$

Wywołanie funkcji w *main()*

```
//  dla f  start 0.5 -> 0     0.8 -> -1.8954  1.0 -> 1.8954
//  dla g  start 0.2 -> 1.5   0.8 ->  ?????   1.0 -> 1.165561
……
newton(&x0, n, eps, f, fp); //f, fp - wskaźniki do funkcji
……
```

```
int newton(double *x, int n, double eps, double(*f1)(double), double(*f2)(double))
{
	double x1;
	int i = 0;
	do
	{
		x1 = -(*f1)(*x) / (*f2)(*x);
		*x += x1;
		if (fabs(x1) < eps) return 0;
	} while (i++ < n);
	return 1;
	}
```

5. **Synonimy nazw typów danych.**
W niektórych sytuacjach skomplikowane nazwy typów mogą zaciemniać obraz tworzonego projektu - język C dostarcza słowa kluczowego `typedef`, które pozwala na tworzenie nazw alternatywnych (synonimów) w stosunku do już istniejących. Łatwo sobie wyobrazić sytuację, w której na początku pisania kodu możemy nie być pewni z jakiego typu danych chcemy korzystać (prosty przykład: nie wiemy czy nasze obliczenia powinny być pojedynczej czy podwójnej precyzji) - w takiej sytuacji warto skorzystać z powyższego mechanizmu.

```
typedef int data;
data x; //zmienna x typu int
```

6. **Struktura** - złożony typ grupujący rożnego rodzaju dane w jednym obszarze pamięci. Struktury języka C są namiastką klas i obiektów wieloparadygmatowego języka C++. Składowe struktury opisane są za pomocą typu i nazwy (np. `int zmienna`).

```
/*Deklaracja*/
struct student
{
	char imie[24];
	char *nazwisko;
	int rok;
}x, y;

/*Definicja*/
struct student z;
strcpy(z.imie, "Piotr\0");
z.nazwisko = (char*)malloc(sizeof(char) * 6);
strcpy(z.nazwisko,"Nowak\0");
z.rok = 1999;
```

7. Dostęp do składowych struktur:

a) bezpośredni – `nazwa_struktury.składowa` – do elementów struktury odwołujemy się poprzez kropkę (sposób definicji I, II, III);

```
x.nazwisko;
x.rok;
```

b) pośredni – `wskaznik->składowa` – do elementów struktury odwołujemy się poprzez operator-> (sposób definicji IV);

```
xx->nazwisko;
xx->rok;
```

Uwaga:

```
x.rok    ≡    (*xx).rok    ≡    xx->rok
```

8. Instancje struktur można definiować na kilka sposobów:

c) Sposób I – aby zdefiniować strukturę z o tych samych składowych, trzeba ponownie wymienić wszystkie składowe;

```
struct
{
        char imie[24];
        char *nazwisko;
        int rok;
}x, y;
```

d) Sposób II - w sposób domyślny parą typ-zmienna, identyfikator `student` nazywany jest etykietą struktury;

```
struct student              /*deklaracja*/
{
        char imie[24];
        char *nazwisko;
        int rok;
};
struct student x,y;         /*definicja*/
```

e) Sposób III – definiowany jest nowy typ `STUDENT`, etykieta `student` może być pominięta;

```
typedef struct student
{
        char imie[24];
        char *nazwisko;
        int rok;
} STUDENT;
STUDENT x, y; //lub struct student x,y;
```

f) Sposób IV - poprzez wskaźnik.

```
STUDENT *xx;
xx = &x;
```

9. Struktury i wskaźniki do struktur można gromadzić w tablicach:

```
STUDENT t[10], *s[5]; //t -tabl. struktur, s - tabl. wskaźników do struktur
t[0];         //struktura o indeksie 0 typu STUDENT
t[0].rok;     //składnik rok struktury o indeksie 0 - int
t[0].imie[1];//element o indeksie 1 składnika imie struktury o indeksie 0-char

s[0];                //zerowy element tablicy typu STUDENT *
(*s[0]).rok;      //składnik rok osoby wskazywanej przez s[0]
s[0]->rok;        //inny zapis powyższego
s[0]->imie[1];  //druga litera imienia osoby wskazywanej przez s[0]
```

10. Napisz funkcje wykonujące operacje na liczbach zespolonych:

1. Funkcję obliczającą pierwiastki (rzeczywiste lub zespolone) równania kwadratowego.
2. Funkcję dodającą dwie liczby zespolone

Zadanie zrealizować definiując strukturę o dwóch składowych typu double reprezentujących odpowiednio część rzeczywistą i część urojoną liczby zespolonej.

```
typedef struct CO {
        double re;
        double im;
} COMPLEX;
```

Przypomnienie. Dla Δ<0 rozwiązania mają postać:

$$x_{re}^{(1)} = -\frac{b}{2a} \qquad x_{im}^{(1)} = +\sqrt{\left|\left(\frac{b}{2a}\right)^2 - \frac{c}{a}\right|}$$

$$x_{re}^{(2)} = -\frac{b}{2a} \qquad x_{im}^{(2)} = -\sqrt{\left|\left(\frac{b}{2a}\right)^2 - \frac{c}{a}\right|}$$

11. W pliku tekstowym zapisane są następujące dane o wielu studentach:

imię, nazwisko, rok urodzenia, adres zamieszkania, kwota stypendium.

```
struct student {
        char* imie;
        char* nazwisko;
        int rok;
        char* adres;
        double stypendium;
};
```

Zapisz te dane do tablicy struktur i wybierz spośród wszystkich osób tego studenta który pobiera największe stypendium.

Uporządkuj tablicę struktur malejąco wg kwoty stypendium.