Politechnika Wrocławska

Programowanie proceduralne
Powtórzenie

Sylwia Majchrowska

Treści programowe

1. Funkcje i struktura programu.
2. Algorytmy sortowania i przeszukiwania tablic.
3. Definiowanie struktur danych. Wskaźniki do struktur.
4. Wejście i wyjście. Odczyt i zapis do plików.

Wymagania wstępne

- Zna składnie i podstawowe instrukcje strukturalnego języka programowania.
- Potrafi sformułować proste algorytmy.
- Potrafi skompilować i uruchomić program napisany w wybranym języku programowania.
- Potrafi zaimplementować proste algorytmy w wybranym języku programowania.

Wymagane treści programowe

1. Zapoznanie z wybranym środowiskiem programistycznym, uruchomienie pierwszych programów.
2. Objaśnianie kodu za pomocą komentarzy. Pisanie na ekran. Zmienne i zmienne łańcuchowe.
3. Dyrektywy #include i #define. Interakcja z użytkownikiem.
4. Obliczenia matematyczne. Instrukcja przypisania.
5. Sterowanie – instrukcja if-else. Operatory logiczne.
6. Sterowanie – pętle for, while, do-while; instrukcja switch.
7. Wykorzystanie łańcuchów znakowych. Funkcje matematyczne.
8. Tablice. Tablice i wskaźniki.

Cele przedmiotu

- Rozszerzenie wiedzy z zakresu składni i instrukcji strukturalnego języka programowania.
- Nabycie umiejętności formułowania i implementacji algorytmów wykorzystujących: funkcje, rekurencję i iterację, różne struktury danych.
- Nabycie umiejętności wykorzystywania poznanych algorytmów (w tym algorytmów sortowania).

Funkcje i struktura programu

Deklaracja, definicja, wywołanie

Deklaracja:

```
<typ_rezultatu> nazwa_funkcji ( lista_parametrów_formalnych );
```

Definicja:

```
<typ_rezultatu> nazwa_funkcji ( lista_parametrów_formalnych )
{
    ciało_funkcji
}
```

Dozwolone wersje poprawnej struktury programu:

```
. . .
int przywitanie( void );

int main()
{
  przywitanie();

  return EXIT_SUCCESS;
}

int przywitanie( void )
{
  . . .
}
```

```
. . .

int przywitanie( void )
{
  . . .
}

int main()
{
  przywitanie();

  return EXIT_SUCCESS;
}
```

Funkcje i struktura programu

Struktura programu

Ogólna **struktura** programu w C / C++

Nagłówek programu / komentarze */*

#include *(włączenia tekstowe - preprocesor)*

#define stałe makroinstrukcje

Zmienne globalne

Prototypy funkcji

Funkcja main()

Funkcje pozostałe

```
/* Przykładowy program P02c.c */
#include <stdio.h> /* preprocesor - załączenie pliku bibliotecznego stdio.h */
#include <conio.h>  /* załącz. pliku do clrscr() i getch() */
#define PI 3.14159  /* definicja PI */
#define P(a) a*a  /* definicja funkcji P(a) jako kwadrat liczby a */
int main(void)  /* funkcja główna */
{   /* klamra otwierająca funkcji głównej*/
float r; /* deklaracja zmiennej rzeczywistej r */
clrscr(); /* czyszczenie ekranu */
puts("Obliczenie pola kola"); /* wyświetlenie łańcucha */
printf("Podaj promien kola: "); /* wyswietl. napisu */
scanf("%f",&r); /* wczytanie adresu promienia r */
printf("PI=%f\n",PI); /* wyświetlenie PI */
printf("Pole kola o promieniu %f = %10.4f\n", r, PI*P(r)); /*wydruk format.: r (całkowita. szer. 10, po kropce 2), pole, po kropce 4 miejsca), \n – nowa linia */
printf("Obwod  = %6.3f \n",2*PI*r); /* wydruk obwodu */
puts("Nacisnij cos"); /* wyświetlenie łańcucha */
getch(); /* Czekanie na naciśniecie klawisza */
return 0;  /* funkcja main() zwraca 0 * - ostatnia przed } */
}  /* klamra zamykająca  funkcji głównej */
```

Funkcje i struktura programu

Sposoby przekazywania argumentów

Przez wartość

```
void inc( int a )
{
  a++;
}

int main()
{
  int i = 5;

  inc( i );

  . . .

}
```

vs

Przez wskaźnik

```
void inc( int *a )
{
  a++;
}

int main()
{
  int i = 5;

  inc(&i );

  . . .

}
```

W języku C argumenty przekazywane są przez:

- wartość
- wskaźnik

Tablice → `void inc( int a[])`

Nieobecność jakiejkolwiek wartości

```
void fun( void )
{
  . . .
}
```

Zapis równoważny – zwracanie liczby całkowitej

```
int fun()
{
  . . .
}
```

```
fun()
{
  . . .
}
```

Zapis równoważny – niezdefiniowana liczba argumentów

```
int fun()
{
  . . .
}
```

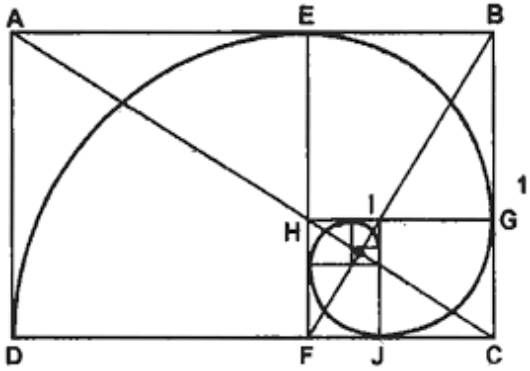
```
int fun( ... )
{
  . . .
}
```

Funkcje i struktura programu

Definicje rekurencyjne i iteracyjne

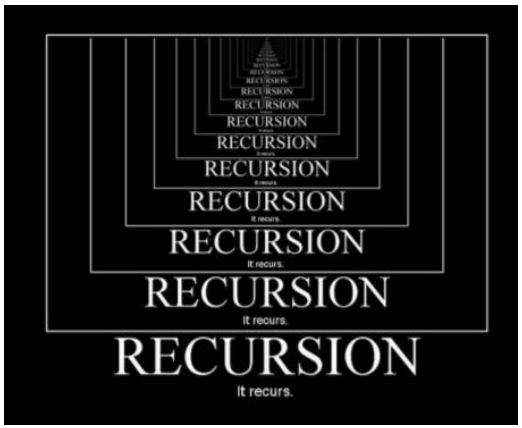
Rekurencja często łatwiejsza w zapisaniu niż iteracja.

```
long long fibonacciR(int n)
{
  if(n<3)
    return 1;

  return fibonacciR(n-2)+fibonacciR(n-1);
}
```

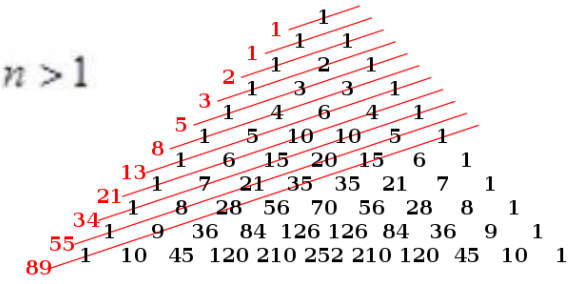


Iteracja jest na ogół bardziej efektywna obliczeniowo niż rekurencja.

```
long long fibonacciI(int n)
{
    long long a = 0, b = 1;

    for(int i=1;i<n;i++)
    {
        b += a;
        a = b-a;
    }
    return b;
}
```



$$F(n) = \begin{cases} 0 & \text{gdy } n = 0 \\ 1 & \text{gdy } n = 1 \\ F(n-1) + F(n-2) & \text{gdy } n > 1 \end{cases}$$



Funkcje i struktura programu

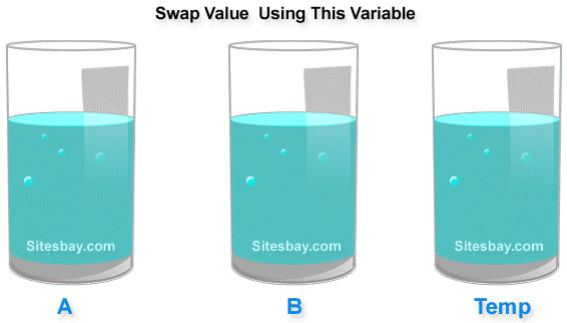
Pytania kontrolne

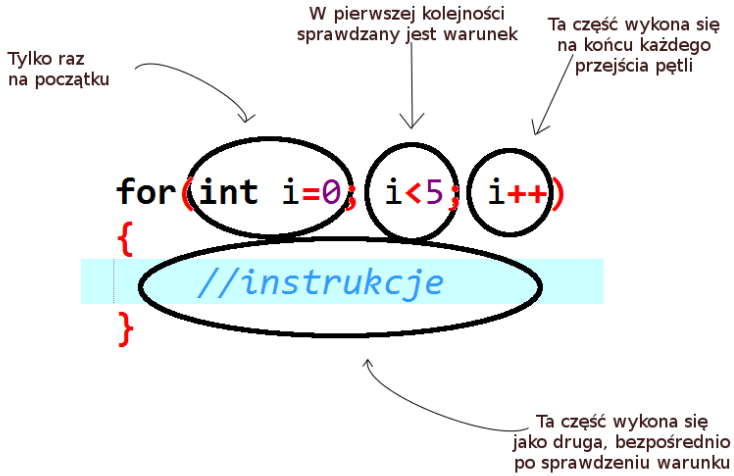
1. Jak wygląda przykładowa struktura programu? (W którym miejscu załączane są biblioteki, umieszczane deklaracje, definicje i wywołanie dowolnej funkcji?)
2. Jak będzie wyglądała deklaracja funkcji zwracającej zmienną zmiennoprzecinkową i przyjmującej dwa argumenty całkowitoliczbowe?
3. Czym różnią się od siebie: deklaracja i definicja funkcji?
4. Do czego służy instrukcja return?
5. Jakie zasięg mogą mieć utworzone zmienne?
6. Jakie znasz sposoby przekazywania argumentów do funkcji?
7. W jaki sposób przekazywany jest argument do funkcji scanf()?
8. Co zwyczajowo zwraca główna funkcja programu – main()?
9. W jaki sposób przekazujemy tablice do funkcji?
10. Czym jest rekurencyjna definicja funkcji?
11. Na czym polega iteracyjna definicja funkcji?
12. Jak będzie wyglądała deklaracja funkcji nie zwracającej nic (jedynie wykonującej pewne operacje) i przyjmującej jako argument tablicę?
13. W jaki sposób i w którym miejscu programu wywołujemy zdefiniowane przez nas funkcje?

Algorytmy sortowania i przeszukiwania tablic
Podstawowe zagadnienia

```
void fbubblesort(float *tab, int n);
void fbubblesort(float tab[], int n);
```

```
void fswap(float *tab, int l, int r)
    float tmp = tab[l];
    tab[l] = tab[r];
    tab[r] = tmp;
```



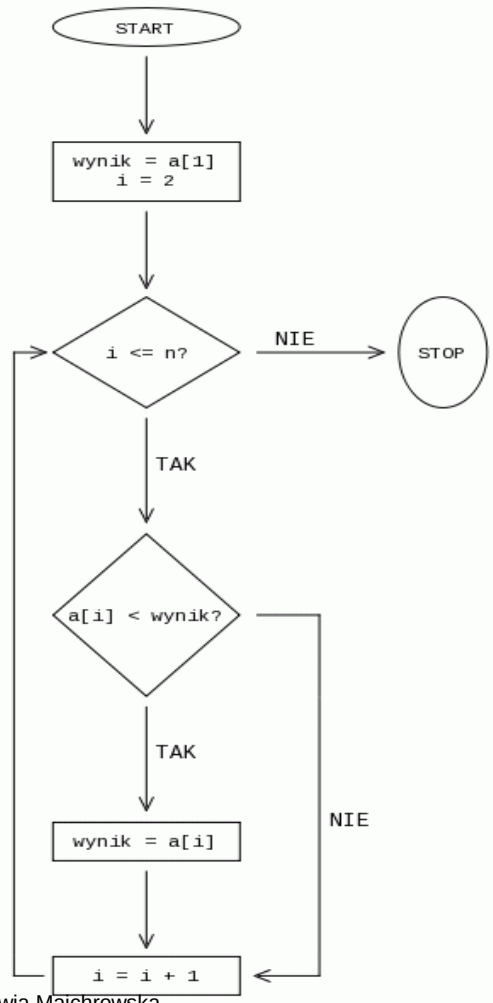



```
float tablica[N] = {3,2.5,4,56.8,12.6};
for(i = 0; i < N; ++i){
    scanf("%.f ",&tablica[i]);
}
```

```
for(i = 0; i < N; ++i){
    printf("%.f ",tablica[i]);
}
```

Algorytmy sortowania i przeszukiwania tablic

Znajdowanie szczególnych elementów i ich indeksów




```
float el_min(float *tab, int n) {
    return tab[ind_el_min(tab, n)];
}

int ind_el_min(float *tab, int n) {
    int i, kand = 0;
    for (i = 1; i < n; i++)
        if (tab[i] < tab[kand])
            kand = i;
    return kand;
}
```

```
float el_max(float *tab, int n){
    return tab[ind_el_max(tab, n)];
}

int ind_el_max(float *tab, int n) {
    int i, kand = 0;
    for (i = 1; i < n; i++)
        if (tab[i] > tab[kand])
            kand = i;
    return kand;
}
```

Algorytmy sortowania i przeszukiwania tablic
Algorytmy sortowania

Iteracyjnie		Rekurencyjnie	
Sortowanie bąbelkowe	1) Kolejno porównaj sąsiednie liczby i zamień je miejscami, jeśli są w złej 2) Po przejściu przez tablicę, wykonaj kolejną serię sprawdzeń dla mniejszego podciągu	Sortowanie przez scalanie	1) Podziel zestaw danych na dwie równe części 2) Zastosuj sortowanie przez scalanie dla każdej z nich oddzielnie, chyba że pozostał już tylko jeden element 3) Połącz posortowane podciągi w jeden ciąg posortowany
Sortowanie przez wybór	1) Wyszukaj minimalną wartość z tablicy spośród elementów od i do końca tablicy 2) Zamień wartość minimalną, z elementem na pozycji i	Sortowanie szybkie	1) Podziel zbiór na dwie części w taki sposób, aby wszystkie elementy leżące w pierwszej były mniejsze lub równe od wszystkich elementów 2) Zastosuj sortowanie szybkie dla każdej części, chyba że pozostał już tylko jeden element 3) Połącz posortowane podciągi w jeden ciąg posortowany

Algorytmy sortowania i przeszukiwania tablic
Pytania kontrolne

1. W jaki sposób (opisz algorytm) znajdziesz minimalny element w tablicy?
2. Na czym polega (ogólna idea) sortowanie przez scalanie (*mergesort*)?
3. Czym jest algorytm? W jaki sposób może zostać zapisany?
4. Jaka jest budowa (struktura) pliku źródłowego (rozszerzenie *.c)?
5. Na czym polega sortowanie przez wybór (*selectsort*)?
6. Jaka jest budowa (struktura) pliku nagłówkowego (rozszerzenie *.h)?
7. W jaki sposób (opisz algorytm) znaleźć indeks odpowiadający elementowi maksymalnemu?
8. Jakie znasz algorytmy sortowania elementów w tablicy?

Struktury danych. Wskaźniki do struktur
Definiowanie struktur

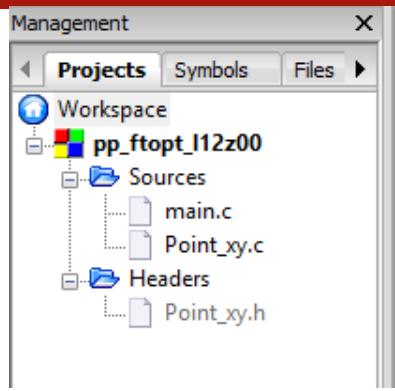
```
struct [etykieta struktury] {
    definicja składowej;
    definicja składowej;
    ...
    definicja składowej;
};
```

```
typedef struct [etykieta struktury] [nowa nazwa]
```

```c
struct Punkt_xy{
    float x;
    float y;
    int cwiartka;
};

int wyznacz_cwiartke(float x, float y);
void readPoint_xy(struct Punkt_xy* p);
void printPoint_xy(struct Punkt_xy p);
int podaj_cwiartke(struct Punkt_xy p);
```

```c
void readPoint_xy(struct Punkt_xy* p){
    printf("Wspolrzedna x: ");
    scanf("%f",&(p->x));

    printf("Wspolrzedna y: ");
    scanf("%f",&(p->y));

    (p->cwiartka) = wyznacz_cwiartke(p->x, p->y);
}

void printPoint_xy(struct Punkt_xy p){
    printf("[%.1f,%.1f]\n",p.x,p.y);
}
```

Struktury danych. Wskaźniki do struktur
Tworzenie projektów



Point_xy.h | Point_xy.c | main.c

```c
#include "Point_xy.h"

int main(){
    struct Punkt_xy punkt1;
    struct Punkt_rfi punkt2;

    readPoint_xy(&punkt1);
    printf("Punkt znajduje sie w %d. cwiartce.\n", podaj_cwiartke(punkt1));

    punkt2 = convert_to_polar(punkt1);
```

Point_xy.h

```c
#ifndef POINT_XY_H_INCLUDED
#define POINT_XY_H_INCLUDED

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

struct Punkt_xy{
    float x;
    float y;
    int cwiartka;
};

int wyznacz_cwiartke(float x, float y);
void readPoint_xy(struct Punkt_xy* p);
void printPoint_xy(struct Punkt_xy p);
int podaj_cwiartke(struct Punkt_xy p);

#endif // POINT_XY_H_INCLUDED
```

Point_xy.h | Point_xy.c | main.c

```c
#include "Point_xy.h"

int wyznacz_cwiartke(float x, float y){
    if(x == 0 || y == 0)
        return 0;
    if(x > 0 && y > 0)
        return 1;
    if(x < 0 && y < 0)
        return 2;
    if(x < 0 && y < 0)
        return 3;
    if(x > 0 && y < 0)
        return 4;
}

void readPoint_xy(struct Punkt_xy* p){
    printf("Wspolrzedna x: ");
    scanf("%f",&(p->x));

    printf("Wspolrzedna y: ");
    scanf("%f",&(p->y));

    (p->cwiartka) = wyznacz_cwiartke(p->x, p->y);
}

void printPoint_xy(struct Punkt_xy* p){
```

Struktury danych. Wskaźniki do struktur
Pytania kontrolne

1. Do czego służą struktury? Czym różnią się one od tablic?
2. Do czego w przypadku struktur służy operator ., a do czego ->?
3. Jak wygląda definicja struktury i w którym miejscu w kodzie się ją umieszcza?
4. Do czego służy instrukcja *typedef*?
5. Jak zadeklarować tablicę struktur?

Wejście i wyjście. Odczyt i zapis do plików

Praca z plikiem w trybie odczytu, zapisu, dołączenia

```c
#define N 50

int main()
{
    srand(time(NULL));
    int i, n;
    char nazwa[N];

    printf("Podaj liczbe n: ");
    scanf("%d", &n);
    //deklaracja wskaznika do pliku
    FILE* fptr;
    printf("Podaj nazwe pliku: ");
    scanf("%s", nazwa);

    //otworzenie pliku do zapisu
    fptr = fopen(nazwa,"w");
    //zapis danych sformatowanych do pliku
    //(tylko jesli plik zostal poprawnie otwarty)
    if( fptr != NULL ){
        for(i=0; i < n; ++i){
            fprintf(fptr,"%.3f\t",rand()/(float)RAND_MAX);
        }
        fclose(fptr);

    } else {
        printf("Plik niepoprawny!");
    }
    return 0;
}
```

Załączenie bibliotek

Tworzymy wskaźnik na plik

Otwarcie w odpowiednim trybie: r, w, a…

Zapis do/odczyt z pliku (tutaj zapis)

Zamknięcie pliku

Wejście i wyjście. Odczyt i zapis do plików
Formatowanie tekstu

Flagi:

- (minus) oznacza, że pole ma być wyrównane do lewej, a nie do prawej.

+ (plus) oznacza, że dane liczbowe zawsze poprzedzone są znakiem (plusem dla liczb nieujemnych lub minusem dla ujemnych).

spacja oznacza, że liczby nieujemne poprzedzone są dodatkową spacją; jeżeli flaga plus i spacja są użyte jednocześnie to spacja jest ignorowana.

Szerokość pola i precyzja:

Minimalna szerokość pola oznacza ile najmniej znaków ma zająć dane pole. Jeżeli wartość po formatowaniu zajmuje mniej miejsca jest ona wyrównywana spacjami z lewej strony

Format:

%d, **%i** - argument typu int jest przedstawiany jako liczba całkowita

%f, **%F** - argument typu float jest przedstawiany jako liczba rzeczywista

%e, **%E** - argument typu float w formacie naukowym

%g, **%G** - argument typu float w formacie naukowym lub zwykłym

%c - argument typu unsigned char

%s - argument typu wskaźnik na char (łańcuch znaków)

Wejście i wyjście. Odczyt i zapis do plików
Pytania kontrolne

1. Do czego służy i jak należy wywoływać funkcję **fopen()** z biblioteki **stdio**?
2. W jaki sposób zamykamy otwarty plik?
3. Jakie są podstawowe tryby dostępu do pliku?
4. Co się stanie jeśli spróbujemy otworzyć nieistniejący plik?
5. Do czego służy i jak należy wywoływać funkcję **fprintf()**?
6. Za pomocą jakiej funkcji wczytujemy dane z pliku, jak wygląda jej wywołanie?
7. Co możemy wpisać pomiędzy znacznikiem **%**, a znakiem określającym typ danych w przypadku funkcji **printf()**?
8. Co zwraca funkcja **scanf()**? Podaj typ oraz interpretację zwracanej wartości.

Przedmiotowe efekty kształcenia

Z zakresu wiedzy:

- Zna składnie i instrukcje strukturalnego języka programowania.
- Zna podstawowe algorytmy sortowania.

Przedmiotowe efekty kształcenia

Z zakresu umiejętności:

- Potrafi korzystać z różnych struktur danych we własnych implementacjach.
- Potrafi implementować podane algorytmy w wybranym języku programowania, uwzględniając funkcje.
- Potrafi implementować algorytmy wykorzystujące funkcje biblioteczne.

Przedmiotowe efekty kształcenia

Z zakresu kompetencji:

- Rozumie potrzebę samodzielnego zdobywania wiedzy.

WYDZIAŁ ...*PPT*... / STUDIUM.....

KARTA PRZEDMIOTU

Nazwa w języku polskim ...*Programowanie proceduralne*
Nazwa w języku angielskim ...*Programing* .....
Kierunek studiów (jeśli dotyczy): ... *Fizyka techniczna* .....
Specjalność (jeśli dotyczy): .....
Stopień studiów i forma: I /~~II stopień~~*, stacjonarna /~~niestacjonarna*~~
Rodzaj przedmiotu: obowiązkowy /~~wybieralny~~ /~~ogólnouczelniany~~ *
Kod przedmiotu .....
Grupa kursów TAK /~~NIE~~*

	Wykład	Ćwiczenia	Laboratorium	Projekt	Seminarium
Liczba godzin zajęć zorganizowanych w Uczelni (ZZU)	15		30		
Liczba godzin całkowitego nakładu pracy studenta (CNPS)	30		60		

Literatura

1. G. Perry, D. Miller, Język C Programowanie dla początkujących
2. B. W. Kernighan, D. M. Ritchie, Język ANSI C Programowanie
3. P. Mikołajczak, Język C – podstawy programowania
4. Notatki z wykładu oraz laboratorium *Programowanie proceduralne* semestr letni 2018/19
5. Karty przedmiotów