

Wstęp do programowania w języku C

Marek Piotrów - Wykład 7
Struktury i wskaźniki do struktur i funkcji. Sterta.

19 listopada 2014

Struktury - punkty.h

```
/****** OPERACJE NA PUNKTACH I WEKTORACH *****/
```

```
struct punkt {  
    float x;  
    float y;  
};
```

```
struct wektor {  
    float x;  
    float y;  
};
```

```
/****** PROTOTYPY FUNKCJI *****/
```

```
struct wektor punkty_na_wektor(struct punkt p0, struct punkt p1);  
void dodaj_wektor(struct punkt *p, struct wektor w);  
struct wektor prostopadly(struct wektor w);  
struct wektor mnoz_przez_liczbe(struct wektor w, float a);  
struct punkt trojkat_rb(struct punkt p1, struct punkt p2);  
double dlugosc(struct wektor w);
```

Struktury - szesciokat.c

```
#include <stdio.h>
#include "punkty.h"

/***** WYZNACZANIE WSPOLRZEDNYCH SZESCIOKATA FOREMNEGO *****/
/*
 * Ten program należy kompilować łącznie z modulem punkty.c
 * np. poleceniem: gcc -std=c99 -o szesciokat szesciokat.c punkty.c -lm
 * lub tworząc odpowiedni projekt w Code::Blocks
 */

int main(void)
{
    struct punkt p1,p2,ps,pp,pn;
    int i;

    printf("Podaj współrzędne dwóch wierzchołków szesciokata foremego\n");
    printf("x1 y1 = "); scanf("%f %f",&p1.x,&p1.y);
    printf("x2 y2 = "); scanf("%f %f",&p2.x,&p2.y);
    ps=trojkat_rb(p1,p2);
    for (i=3,pp=p2; i <= 6; ++i,pp=pn) {
        pn=trojkat_rb(ps,pp);
        printf("%i-%s wierzchołek ma współrzędne (x%i,y%i) = (%.2f,%.2f)\n",
            i,(i==3 ? "ci" : "ty"),i,i,pn.x,pn.y);
    }
    return 0;
}
```

Struktury - punkty.c I

```
#include <math.h>
#include "punkty.h"
```

/ Moduł PUNKTY implementujący operacje na punktach i wektorach 2-wymiarowych. */*

```
struct wektor punkty_na_wektor(struct punkt p0, struct punkt p1)
{
    struct wektor w;

    w.x = p1.x - p0.x;
    w.y = p1.y - p0.y;
    return w;
}
```

```
void dodaj_wektor(struct punkt *p, struct wektor w)
{
    p->x += w.x;
    p->y += w.y;
}
```

```
struct wektor prostopadly(struct wektor w)
{
    struct wektor wp = { .x = -w.y, .y = w.x };

    return wp;
}
```

Struktury - punkty.c II

```
struct wektor mnoz_przez_liczbe(struct wektor w, float a)
{
    w.x*=a;
    w.y*=a;
    return w;
}
```

```
struct punkt trojkat_rb(struct punkt p1, struct punkt p2)
{
    struct wektor w12, w14, w43;
    struct punkt p3 = p1;

    w12=punkty_na_wektor(p1, p2);
    w14=mnoz_przez_liczbe(w12, 0.5);
    w43=mnoz_przez_liczbe(prostopadly(w14), sqrt(3.0));
    dodaj_wektor(&p3, w14);
    dodaj_wektor(&p3, w43);
    return p3;
}
```

```
double dlugosc(struct wektor w)
{
    return sqrt(w.x*w.x+w.y*w.y);
}
```

Wskaźniki do struktur - odlegosc.c

```
#include <stdio.h>
#include <string.h>
#include "punkty.h"
// Ten program musi byc kompilowany łącznie z modulem punkty.c

#define MAXDL 20
#define MAX 300

struct n_punkt {
    char nazwa[MAXDL];
    struct punkt polozenie;
};

/***** WYZNACZANIE ODLEGLOSCI MIEDZY MIASTAMI *****/

static int wczytaj_miasta(struct n_punkt miasto[]);
static struct n_punkt *szukaj(struct n_punkt miasto[], int liczba, char *nazwa);
```

odleglosc.c - cd. - funkcja main

```
int main(int argc, char *argv[])
{
    struct n_punkt miasta[MAX];
    int liczba_miast=0;
    char *miasto1, *miasto2, *m;
    struct n_punkt *p1, *p2;
    double odleglosc;

    liczba_miast=wczytaj_miasta(miasta);
    if (argc < 3 || liczba_miast == 0) {
        printf("uzycie: %s miasto1 miasto2 <baza.txt\n", argv[0]);
        return(1);
    }
    miasto1=argv[1];
    miasto2=argv[2];
    if ((p1=szukaj(miasta, liczba_miast, m=miasto1)) == NULL ||
        (p2=szukaj(miasta, liczba_miast, m=miasto2)) == NULL) {
        printf("Brak miasta w bazie: %s\n", m);
        return(2);
    }
    odleglosc=dlugosc(punkty_na_wektor(p1->polozenie, p2->polozenie));
    printf("Odleglosc miedzy %s a %s wynosi %.2lf km\n",
           miasto1, miasto2, odleglosc);
    return 0;
}
```

odleglosc.c cd. - funkcje pomocnicze

```
static int wczytaj_miasta(struct n_punkt miasto[])
{
    int i=0,res;

    do
        res=scanf("%s %f %f",miasto[i].nazwa,&miasto[i].polozenie.x,&miasto[i].polozenie.y);
    while (res == 3 && ++i < MAX);
    return i;
}
```

```
static struct n_punkt *szukaj(struct n_punkt miasto[],int liczba,char *nazwa)
{
    int porownanie;
    struct n_punkt *dol=&miasto[0],*gora=&miasto[liczba],*srodek;

    while (dol < gora) {
        srodek=dol+(gora-dol)/2;
        if ((porownanie=strcmp(nazwa,srodek->nazwa)) < 0)
            gora=srodek;
        else if (porownanie > 0)
            dol=srodek+1;
        else
            return srodek;
    }
    return NULL;
}
```


Nazwy tablic i funkcji jako stałe wskaźnikowe

- Identyfikator tablicy zadeklarowanej jako:

`<typ> tab[...]`

jest w większości przypadków traktowany jak stała wskaźnikowa typu:

`<typ> *.`

- Podobnie, identyfikator funkcji zadeklarowanej jako:

`<typ> fun(...)`

jest traktowany jak stała wskaźnikowa typu:

`<typ> (*) (...).`

Wykresy funkcji - wskaźniki do funkcji

```
#include <stdio.h>
#include <math.h>
#include <string.h>
// Program rysuje wykres jednej z sześciu poniższych funkcji w zadanym przedziale.

#define ILOSCF 6

static char *nazwaf[ILOSCF]={"sin","cos","sqrt","exp","log","sinh"};
static double (*funkcja[ILOSCF])(double x) = {sin,cos,sqrt,exp,log,sinh};

#define MAXX 80
#define MAXY 20

static char wykres[MAXY][MAXX+1];

static void wyznacz_wykres(double (*funkcja)(double x),double a,double b,
    double *miny,double *maxy);
static void rysuj_wykres(char *nazwaf,double a,double b,double miny,double maxy);
```

Wykresy funkcji - funkcja `main`

```
int main(void)
{
    int nr;
    double a,b,miny,maxy;

    printf("Rysowanie wykresu wybranej funkcji w zadanym przedziale:\n");
    for (int i=0; i < ILOSCF; ++i)
        printf("%3i) %5s(x)\n",i+1,nazwaf[i]);
    do {
        printf("Wybierz numer funkcji (1-%i) : ",ILOSCF);
        scanf("%d",&nr);
    } while (nr < 1 || nr > ILOSCF);
    do {
        printf("Podaj przedzial [a,b] dla x : ");
        scanf("%lf %lf",&a,&b);
    } while (a >= b);
    wyznacz_wykres(funkcja[nr-1],a,b,&miny,&maxy);
    rysuj_wykres(nazwaf[nr-1],a,b,miny,maxy);
    return 0;
}
```

```
static void wyznacz_wykres(double (*funkcja)(double x), double a, double b,
                           double *miny, double *maxy)
{
    double x, krokx, kroky, minw, maxw;
    double wart[MAXX];

    krokx = (b - a) / (MAXX - 1); x = a + krokx;
    wart[0] = minw = maxw = (*funkcja)(a);
    for (int i = 1; i < MAXX; ++i, x += krokx)
        if ((wart[i] = (*funkcja)(x)) > maxw)
            maxw = wart[i];
        else if (wart[i] < minw)
            minw = wart[i];
    kroky = (maxw - minw) / (MAXY - 1);

    memset(wykres, ' ', sizeof(wykres));
    for (int i = 0; i < MAXY; ++i)
        wykres[i][MAXX] = ' \0 ';
    if (a <= 0.0 && 0.0 <= b)
        for (int i = 0, j = (int)nearbyint(-a / krokx); i < MAXY; ++i)
            wykres[i][j] = ' | ';
    if (minw <= 0.0 && 0.0 <= maxw)
        for (int i = (int)nearbyint(-minw / kroky), j = 0; j < MAXX; ++j)
            wykres[i][j] = (wykres[i][j] == ' ' ? ' - ' : ' + ');

    for (int j = 0; j < MAXX; ++j)
        wykres[(int)nearbyint((wart[j] - minw) / kroky)][j] = ' * ' ;

    *miny = minw; *maxy = maxw;
    return;
}
```

Wykresy funkcji - rysowanie wykresu

```
static void rysuj_wykres(char *nazwaf,double a,double b,double miny,double maxy)
{
    printf("===== Wykres funkcji %s(x) dla x z przedzialu [%.2lf,%.2lf] "
           "=====\n",nazwaf,a,b);
    for (int i=MAXY-1; i >= 0; --i)
        printf("%s\n",wykres[i]);
    printf("===== Wartosci funkcji na osi Y sa z przedzialu [%.2lf,%.2lf] "
           "=====\n",miny,maxy);
}
```

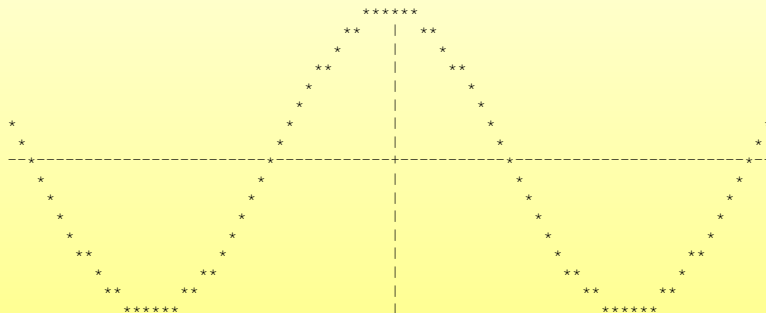
Rysowanie wykresu wybranej funkcji w zadanym przedziale:

- 1) $\sin(x)$
- 2) $\cos(x)$
- 3) \sqrt{x}
- 4) $\exp(x)$
- 5) $\log(x)$
- 6) $\sinh(x)$

Wybierz numer funkcji (1-6) : 2

Podaj przedział [a,b] dla x : -5 5

===== Wykres funkcji $\cos(x)$ dla x z przedziału $[-5.00, 5.00]$ =====



===== Wartości funkcji na osi Y są z przedziału $[-1.00, 1.00]$ =====

Skomplikowane deklaracje - przykłady

```
char **argv;           /* wskaźnik do wskaźnika do typu char */

int (*daytab)[13];      /* wskaźnik do tablicy [13] o elementach typu int */

void *comp();           /* funkcja zwracająca wskaźnik do typu void */

void (*comp)();         /* wskaźnik do funkcji zwracającej typ void */

char (*(x())[12])();    /* funkcja zwracająca wskaźnik do tablicy [12] o elementach
                           typu wskaźnik do funkcji zwracającej typ char */

char (*(x[3])())[5];    /* tablica [3] o elementach typu wskaźnik do funkcji
                           zwracającej wskaźnik do tablicy [5] o elementach
                           typu char */
```

Deklaracje wskaźników z użyciem `const`

- `const int *wsk_do_stalej;` - wartość wskazywana nie może być modyfikowana za pomocą tego wskaźnika; sam wskaźnik może być modyfikowany.
- `int *const staly_wsk;` - wskaźnik `staly_wsk` nie może być modyfikowany; wartość wskazywana może być zmieniana.
- Można też zadeklarować stały wskaźnik do stałej wartości:
`const int *const wsk;`

Operacje na stercie

- Operacje na stercie nie są częścią definicji języka C (pojawiają się dopiero w definicji C++), ale są dostępne w standardowej bibliotece C `stdlib`.

- Do przydziału pamięci ze sterty używa się jednej z funkcji:

```
void *malloc(size_t size);  
void *calloc(size_t nmemb, size_t size);
```

- Do zwolnienia przydzielonej pamięci służy funkcja:

```
void free(void *ptr);
```

- Do zmiany rozmiaru przydzielonej pamięci można użyć funkcji:

```
void *realloc(void *ptr, size_t size);
```

Struktury dynamiczne - stos.h

```
#include <stdlib.h>
#include <stdbool.h>
/* Interfejs modułu STOS. Typ przechowywanej na stosie wartości określa
 * poniższa definicja */

#define TYP_INFO  char*
#define TYP_NULL  NULL

typedef struct e_stos {
    TYP_INFO info;
    struct e_stos *nast;
} *StosPtr;

void init(StosPtr *stos);
bool isempty(StosPtr stos);
bool push(StosPtr *stos, TYP_INFO info);
TYP_INFO top(StosPtr stos);
TYP_INFO pop(StosPtr *stos);
```

Struktury dynamiczne - stos.c

```
#include <stdlib.h>
#include "stos.h"

void init(StosPtr *stos)
{
    *stos=NULL;
}

bool isempty(StosPtr stos)
{
    return (stos == NULL);
}

bool push(StosPtr *stos, TYP_INFO info)
{
    StosPtr p;

    if ((p=(StosPtr)malloc(sizeof(struct e_stos))) == NULL)
        return false;
    else {
        p->info=info;
        p->nast=*stos;
        *stos=p;
        return true;
    }
}
```

Struktury dynamiczne - stos.c (cd.)

```
TYP_INFO top(StosPtr stos)
{
    return (stos == NULL ? TYP_NULL : stos->info);
}
```

```
TYP_INFO pop(StosPtr *stos)
{
    TYP_INFO info;
    StosPtr p;

    if (*stos == NULL)
        return TYP_NULL;
    else {
        info=(*stos)->info;
        p=*stos;
        *stos=(*stos)->nast;
        free(p);
        return info;
    }
}
```