Podstawy Informatyki

Stanisław Flaga
stanislaw.flaga@agh.edu.pl

Plik binarny – przykład - deklaracje

```c
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>

#define DLUG15 15
#define PLF printf("\n");

bool otworzKontrola(FILE **uchwyt, char *plik, char *tryb);

```

Plik binarny – przykład – funkcja główna - deklaracje

```
11 int main(void){
12     int zapisOdczyt = 0; // TO MUSISZ USTAWIĆ: jeżeli 1 ZAPIS NOWEGO; 0 ODZCZYT
13     bool status;
14     char plik[DLUG15]    = "10-lb-01.dat";    //nazwa otwieranego pliku
15     char trybOp[5];                           //tryb otwarcia pliku
16     int  skalar          = 321;
17     int  wektor[DLUG15]  = {12, 13, 14, 20, 19, 18, 17, 16, 0, 3, 20, 17, 45, 33, 10};
18     int  wektorCzyt[DLUG15], i;
19     size_t przeczytano;
20     FILE * fp;
```

Plik b

```
22    if(zapisOdczyt) strcpy(trybOp, "w+");
23    else strcpy(trybOp, "r+");
24    status = otworzKontrola(&fp, plik, trybOp);
25    if(status){
26      switch(zapisOdczyt){
27        case 0:      //odczyt
28          PLF
29          fseek(fp, 0L, SEEK_SET);
30          przeczytano = fread(wektorCzyt, sizeof (int), 50, fp);
31          printf("\nPrzeczytano: %ld\n", przeczytano);
32          for(i = 0; i < przeczytano; i++)
33          printf("%d, ", wektorCzyt[i]);
34          break;
35        case 1:      //zapis
36          fwrite(&skalar, sizeof (int), 1, fp);
37          fwrite(wektor, sizeof (int), 10, fp);
38          break;
39      }
40    }
41    PLF
42    if(status) fclose(fp);
43    return EXIT_SUCCESS;
44  }
```

Plik binarny – funkcja otwierająca plik (przekazanie wskaźnika do pliku)

```
46 bool otworzKontrola(FILE **uchwyt, char *plik, char *trybOp){
47     //Otwieram plik binarny w zadanym trybie
48     *uchwyt = fopen(plik, trybOp);
49     if(*uchwyt == NULL){
50         printf("\nNie mozna otworzyc pliku: %s\n", plik);
51         exit(EXIT_FAILURE);
52     }
53     return true;
54 }
```

Typ wyliczeniowy przykład

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main(void){
  int wybor;
  char komunikat[30];

  enum status {stSTOP, stOK, stERR, stRUN};

  printf("\n\nswitch z typem wyliczeniowym\n");
  wybor = stOK;
  while(wybor){
    switch(wybor){
      case stSTOP:
        strcpy(komunikat, "STOP");
        break;
      case stOK:
        strcpy(komunikat, "OK");
        break;
      case stERR:
        strcpy(komunikat, "ERR");
        break;
      case stRUN:
        strcpy(komunikat, "RUN");
        break;
      default:
        printf("Wybor spoza zakresu - postaraj sie bardziej\n");
        break;
    }
    printf("\nStatus: %s\nWybierz opcję [0..3]: ", komunikat);
    wybor = (int)getchar() - 48;
    getchar();
    printf(" wybrales: %d\n\n", wybor);
  }
  printf("\n\n");
  return EXIT_SUCCESS;
}
```

Zmienna liczba parametrów przykład

```c
// varargs.c -- uzycie zmiennej liczby argumentow
#include <stdio.h>
#include <stdarg.h>

double sumuj(int, ...);

int main(void)
{
  double s,t;
  s = sumuj(4, 1.0, 2.0, 3.0, 4.0);
  t = sumuj(7, 1.0, 2.0, 3.0, 4.0, 10.0, 20.0, 30.0);
  printf("zwroc wartosc "
  "suma(4: 1, 2, 3, 4): %g\n", s);
  printf("zwroc wartosc "
  "suma(7: 1, 2, 3, 4, 10, 20, 30): %g\n", t);
  return 0;
}
double sumuj(int lim,...)
{
  va_list ap; // deklaracja obiektu przechowujacego argumenty
  double suma = 0;
  int i;
  va_start(ap, lim); // zainicjalizowanie ap lista argumentow
  for (i = 0; i < lim; i++)
    suma += va_arg(ap, double); // dostep do kazdego elementu w liscie
  // argumentow
  va_end(ap); // czyszczenie
return suma;
}
```

Struktura – inicjalizacja oznaczona

```c
#include<stdio.h>
#include<stdlib.h>

int main(void){
    struct rzeczy {int liczba;
                   char kod[4];
                   float cena;
    };
    struct rzeczy cosNiecos = {.liczba = 123,   //inicjalizator oznaczony
                               .kod = "A12",
                               .cena = 10.5
    };

    printf("\n%d\t%s\t%f\n", cosNiecos.liczba, cosNiecos.kod, cosNiecos.cena);
    return EXIT_SUCCESS;
}
```

Tablice struktur - UWAGA

```c
#include<stdio.h>
#include<stdlib.h>
#define MAXRZECZY 100

int main(void){
    int i;
    struct rzeczy {int liczba;
                   char kod[13];
                   float cena;
    };
    struct rzeczy magazyn[MAXRZECZY];

    struct rzeczy cosNiecos = {.liczba = 123,   //inicjalizator oznaczony
                               .kod = "A123456789012",
                               .cena = 10.5
    };
    i = 5;
    magazyn[i] = cosNiecos;
    printf("\n%d\t%s\t%f\n", magazyn[i].liczba, magazyn[i].kod, magazyn[i].cena);
    printf("UWAGA: magazyn[%d].kod[4] = %c\n", i, magazyn[i].kod[4]);

    return EXIT_SUCCESS;
}
```

Tablice struktur

```c
#include<stdio.h>
#include<stdlib.h>
#define MAXRZECZY 100

int main(void){
  int i;
  struct rzeczy {int liczba;
                 char kod[13];
                 float cena;
  };
  struct rzeczy magazyn[MAXRZECZY];

  printf("\nUzupełnianie magazynu: \n");
  i = 0;
  while(i < MAXRZECZY){
    printf("Wprowadz w koejności (separator spacja): liczbe kod cene : rzeczy o indeksie: %d\n", i);
    scanf("%d%s%f", &magazyn[i].liczba, magazyn[i].kod, &magazyn[i].cena);
    i++;
  }

  printf("\n\nWprowadzono\n");
  for(i = 0; i < MAXRZECZY; i++)
    printf("%5d\t%13s\t%7.2f\n", magazyn[i].liczba, magazyn[i].kod, magazyn[i].cena);

  return EXIT_SUCCESS;
}
```

Struktury zagnieżdżone

```c
#include<stdio.h>
#include<stdlib.h>
#define MAXRZECZY 100

int main(void){
  int i;
  struct punkt {float X;
                float Y;
  };

  struct lamana{
    int nrPunktu;
    struct punkt wspolrzedne;
  };

  struct lamana pktStartowy = {
    1,
    {1.25, 0.75}
  };

  printf("\nWypisz wspolrzedne \n");
  printf("\nPunk nr %5d: X = %9.2f; X = %9.2f\n", pktStartowy.nrPunktu,
                                                  pktStartowy.wspolrzedne.X,
                                                  pktStartowy.wspolrzedne.Y);


  return EXIT_SUCCESS;
}
```

Wskaźniki do struktur

```
10
11  struct lamana{
12    int nrPunktu;
13    struct punkt wspolrzedne;
14  } punkt2;
15
16  struct lamana pktStartowy = {
17    1,
18    {1.25, 0.75}
19  };
20
21  struct lamana * wskDoStrukt;
22
23  printf("\nWypisz wspolrzedne \n");
24  printf("\nPunk nr %5d: X = %9.2f;
25
26
27  wskDoStrukt = &punkt2;
```

```
28    wskDoStrukt->nrPunktu = 2;
29    (*wskDoStrukt).wspolrzedne.X = 3.15;
30    wskDoStrukt->wspolrzedne.Y = 5.5;
31
32    printf("\nPunk nr %5d: X = %9.2f; X = %9.2f\n", (*wskDoStrukt).nrPunktu,
33                                                    wskDoStrukt->wspolrzedne.X,
34                                                    (*wskDoStrukt).wspolrzedne.Y);
35
36    return EXIT_SUCCESS;
37  }
```

```c
#include<stdio.h>
#include<stdlib.h>
#define MAXRZECZY 100

struct punkt {float X;
              float Y;
};

struct lamana{
  int nrPunktu;
  struct punkt wspolrzedne;
};

void wypisz(struct lamana mPunkt);

int main(void){
  int i;

  struct lamana punkt2;
  struct lamana pktStartowy = {
    1,
    {1.25, 0.75}
  };

  struct lamana * wskDoStrukt;

  wskDoStrukt = &punkt2;
  wskDoStrukt->nrPunktu = 2;
  (*wskDoStrukt).wspolrzedne.X = 3.15;
  wskDoStrukt->wspolrzedne.Y = 5.5;

  wypisz(pktStartowy);
  wypisz(punkt2);

  return EXIT_SUCCESS;
}

void wypisz(struct lamana mPunkt){
  puts("\nWypisane w funkcji");
  printf("Punk nr %5d: X = %9.2f; X = %9.2f\n", mPunkt.nrPunktu,
                                                mPunkt.wspolrzedne.X,
                                                mPunkt.wspolrzedne.Y);
}
```

```c
#include<stdio.h>
#include<stdlib.h>
#define MAXRZECZY 100

  struct punkt {float X;
                float Y;
  };

  struct lamana{
    int nrPunktu;
    struct punkt wspolrzedne;
  };

void wypisz(struct lamana mPunkt);
void wypiszPrzezWsk(struct lamana * mWskPunkt);
```

```c
int main(void){
  int i;

  struct lamana punkt2;
  struct lamana pktStartowy = {
    1,
    {1.25, 0.75}
  };

  struct lamana * wskDoStrukt;

  wskDoStrukt = &punkt2;
  wskDoStrukt->nrPunktu = 2;
  (*wskDoStrukt).wspolrzedne.X = 3.15;
  wskDoStrukt->wspolrzedne.Y = 5.5;

  wypisz(pktStartowy);
  wypiszPrzezWsk(wskDoStrukt);

  return EXIT_SUCCESS;
}
```

```c
void wypisz(struct lamana mPunkt){
  puts("\nWypisane w funkcji - przekazane przez wartosc");
  printf("Punk nr %5d: X = %9.2f; X = %9.2f\n", mPunkt.nrPunktu,
                                                mPunkt.wspolrzedne.X,
                                                mPunkt.wspolrzedne.Y);
}

void wypiszPrzezWsk(struct lamana * mWskPunkt){
  puts("\nWypisane w funkcji - przekazane przez wskaznik");
  printf("Punk nr %5d: X = %9.2f; X = %9.2f\n", (*mWskPunkt).nrPunktu,
                                                mWskPunkt->wspolrzedne.X,
                                                mWskPunkt->wspolrzedne.Y);
}
```

```c
#include<stdio.h>
#include<stdlib.h>

  struct punkt {float X;
                float Y;
  };

  struct lamana{
    int nrPunktu;
    struct punkt wspolrzedne;
  };

void wypisz(struct lamana Punkt);
struct lamana wprowadz(void);

int main(void){
  int i;
  struct lamana punkt2;

  punkt2 = wprowadz();
  wypisz(punkt2);

  return EXIT_SUCCESS;
}

struct lamana wprowadz(void){
  struct lamana temp;
  printf("Wpisz nr: "); scanf("%d", &temp.nrPunktu);
  printf("Wpisz X: "); scanf("%f",  &temp.wspolrzedne.X);
  printf("Wpisz Y: "); scanf("%f",  &temp.wspolrzedne.Y);
  return temp;
}

void wypisz(struct lamana Punkt){
  puts("\nWypisane w funkcji - przekazane przez wskaznik");
  printf("Punk nr %5d: X = %9.2f; X = %9.2f\n",  Punkt.nrPunktu,
                                                 Punkt.wspolrzedne.X,
                                                 Punkt.wspolrzedne.Y);
}
```