FUNKCJE I STRUKTURA PROGRAMU

Format definicji funkcji

```
typ-powrotu nazwa-funkcji(deklaracje parametrow)
{
  deklaracje i instrukcje
}
```

```
Najmniejsza funkcja

dummy() {}
```

Instrukcja *return* - narzędzie do przekazywania wartości pewnego wyrażenia do miejsca wywołania.

```
return wyrazenie;
```

Jeśli zajdzie potrzeba, wyrażenie zostanie przekształcone do typu wartości zwracanej przez funkcję. Czasem *wyrażenie otacza się nawiasami okrągłymi, co nie jest jednak konieczne.*

Funkcje zwracające wartości niecałkowite

- *Definicja funkcji - przykład*

```
#include <ctype.h>

/* zamiana ciagu s na liczbe zmiennopozycyjna */
double atof(char s[])
{
  double val, power;
  int i,sign;
  for (i=0; isspace(s[i]); i++)
    ;  /* Pomijamy biale znaki */
  sign=(s[i] == '-') ? -1:1;
  if (s[i] == '+' || s[i] == '-')
    i++;
  for (val=0.0; isdigit(s[i]); i++)
    val = 10.0*val +(s[i]-'0');
  if (s[i]=='.')
    i++;
  for (power=1.0; isdigit(s[i]); i++) {
    val=10.0*val + (s[i]-'0');
    power*=10.0;
  }
  return sign*val\power;
}
```

- *Deklaracja funkcji*

```
#include <stdio.h>
#define MAXLINE 100
/*  sumowanie wprowadzanych liczb */
main()
{
  double sum=0.0, atof(char []);
  char line[MAXLINE];
  while (gets(line)>0) {
    if (line[0] == '\0') break;
    printf("\t%g\n",sum+=atof(line));
  }
  return 0;
}
```

Zmienne zewnętrzne

- **zmienne wewnętrzne** - *argumenty i zmienne definiowane wewnątrz funkcji*
- **zmienne zewnętrzne** - *zmienne definiowane poza wszystkimi funkcjami. Istnieją również po wyjściu z funkcji.*

Zasięg nazw

- *Zasięgiem zmiennej automatycznej zadeklarowanej na początku funkcji jest cała funkcja zawierająca deklarację zmiennej.*
- *Zmienne lokalne występujące w różnych funkcjach nie są ze sobą w żaden sposób związane. Parametry funkcji są również zmiennymi lokalnymi.*
- *Zasięg zmiennej zewnętrznej i funkcji zewnętrznej rozciąga się od miejsca zadeklarowania w pliku źródłowym do końca tego pliku.*
- *Jeżeli odwołanie do zmiennej zewnętrznej lub wywołanie funkcji zewnętrznej występuje przed jej definicją lub definicja znajduje się w innym pliku źródłowym niż odwołanie, to należy obowiązkowo użyć deklaracji extern.*

Przykład (Jaki będzie wynik?):

```
#include <stdio.h>
int n=10, q=2;

void main()
{
  int fun(int);
  void f(void);
  int n=0, p=5;
  n=fun(p);
  printf("A: w main, n=%d, p=%d, q=%d\n",n,p,q);
  f();
}

int fun(int p)
{
  int q;
  q=2*p+n;
  printf ("B: w fun,  n=%d, p=%d, q=%d\n", n,p,q);
  return q;
}

void f(void)
{
  int p=q*n;
  printf ("C: w f,    n=%d, p=%d, q=%d\n", n,p,q);
}
```

```
Rozwiazanie:
B: w fun,  n=10, p=5, q=20
A: w main, n=20, p=5, q=2
c: w f,    n=10, p=20, q=2
```

Pliki nagłówkowe

Przykład (program podzielony pomiędzy dwa pliki):

```
 /*   PLIK 1  */
#include <stdio.h>
#include "defs.h"

int n=10, q=2;

void main()
{
```

```
  int n=0, p=5;
  n=fun(p);
  printf("A: w main, n=%d, p=%d, q=%d\n",n,p,q);
  f();
}
```

```
  /*  PLIK 2  */
#include <stdio.h>
extern int n,q;

int fun(int p)
{
  int q;
  q=2*p+n;
  printf ("B: w fun,  n=%d, p=%d, q=%d\n", n,p,q);
  return q;
}

void f(void)
{
  int p=q*n;
  printf ("C: w f,    n=%d, p=%d, q=%d\n", n,p,q);
}
```

```
  /*  Plik defs.h  */
extern int fun(int);
extern void f(void);
```

Zmienne statyczne

Deklaracja static zastosowana do zmiennych i funkcji zewnętrznych ogranicza ich zasięg od miejsca wystąpienia do końca pliku. Jest to więc sposób na ukrycie ich nazw.

```
static char buf[BUFSIZE];
static int bufp=0;
```

Deklaracja static w odniesieniu do zmiennych wewnętrznych powoduje, że nie znikają one pomiędzy kolejnymi wywołaniami funkcji. Nadal pozostają lokalnymi.

Zmienne rejestrowe

Deklaracja register informuje kompilator o tym, że deklarowana zmienna będzie intensywnie używana. Takie zmienne kompilator może umieścić bezpośrednio w rejestrach maszyny.

Zastosowanie – wyłącznie do zmiennych automatycznych i do parametrów formalnych funkcji.

```
register int x;
register char c;

f( register unsigned m, register long n)
{
  register int i;
  ....
}
```

Nie ma możliwości uzyskania adresu zmiennej rejestrowej.

Struktura blokowa

- *Można definiować zmienne wewnątrz bloku (instrukcji złożonej).*
- *Nie wolno definiować funkcji wewnątrz innych funkcji.*
- *Zmienne automatyczne wewnątrz funkcji (łącznie z parametrami) przesłaniają zmienne zewnętrzne i funkcje o tych samych nazwach.*

```
if (n>0) {
  int i; /* def. nowego i */
  for (i=0; i<n; i++)
    .....
}
/* Zasiegiem i jest galaz "prawdy" if */
```

Inicjowanie

- *Jeżeli nie nie podano wartości początkowych, to zmienne zewnętrzne i statyczne są inicjowane zerami*
- *Wartości początkowe zmiennych automatycznych i rejestrowych są nieokreślone.*
- *Wartość początkowa zmiennej zewnętrznej albo statycznej musi być wyrażeniem stałym.*
- *Zmienne automatyczne i rejestrowe są inicjowane przy każdym wejściu do funkcji lub bloku.*

```
Deklaracja tablicy:

int days[]={31,28,31,30,31,30,31,31,30,31,30,31};

dla tablic znakowych mozna uzyc
stalej napisowej:

char patt[]="trawa";
rownowazne
char patt[]={'t','r','a','w','a','\0'};
```

Rekurencja

Wywołanie funkcji wewnątrz tej samej funkcji.
Przykład 1:

```
/*   1  */
int silnia(int n)
{
   int wynik=1;
   while (n--)
     wynik*=n;
   return wynik;
}

/*  2  */
int silnia(int n)
{
   if (n == 1)
     return 1;
   else
     return n*silnia(n-1);
}
```