Opisy funkcji

Adres strony WWW :

http://www.gnu.org/software/libc/manual/html_node/index.html (należy odszukać hyperlink Function Index)

(http://www.gnu.org/manual/manual.html)

http://www.gnu.org/manual/glibc-2.0.6/html_node/libc_528.html

Operatory logiczne

>	większe
>=	większe lub równe
<	mniejsze
<=	mniejsze lub równe
==	równe
!=	nierówne

Operatory logiczne

/* *niech np. Gdzieś wcześniej jest :* int prawda; */

```
if ( !prawda ) instrukcja
```

```
if ( prawda == 0 ) instrukcja
```

(powyższe sposoby są równoważne)

Instrukcja warunkowa

```
if (wyrażenie)
        instrukcja-1
else
        instrukcja-2
```

Instrukcja warunkowa

```
if ( n>0 )
{
        if ( a>b )  z = a;
}
else
        z = b;
```

Instrukcja warunkowa

```
    /* instrukcje warunkowe można zagnieżdżać  */
if ( wyrażenie-1 )
                instrukcja-1
else
if ( wyrażenie-2 )
                instrukcja-2
else
if ( wyrażenie-3 )
                instrukcja-3
else
                instrukcja-4
```

adres zmiennej

Do pobrania adresu zmiennej używa się jednoargumentowego operatora & (uwaga & może mieć także znaczenie dwuargumentowego operatora bitowego iloczynu logicznego)

Jednoargumentowy operator * jest "używany do wskazywania", tzn. jego argument jest adresem zmiennej.

adres zmiennej - przyklad

```
#include <stdio.h>
#include <stdlib.h>
int  main()
{
        int k;
        int n;
        int *palec;
        palec=&k;
        k=10;
        n=(*palec)*15;
        printf("\n\nk=%5d  n=%5d  *palec=%5d\n",k,n,*palec);
        exit(0);
}  /* koniec funkcji main  */
```

```
              k=   10  n=  150  *palec=   10
```

Wskaźniki-0

Czy wskaźnik musi mieć typ ?

Czy może istnieć wskaźnik uniwersalny?

Jeśli tak to:

Czy można się posługiwać tylko wskaźnikami uniwersalnymi?

Wskaźniki-1

```
#include <stdio.h>
int main()
{
   float a[10]={0.,1.,2.,3.,4.,5.,6.,7.,8.,9.};
   float * f1;  /* wskaźnik  do  float */
   int   * p1;  /* wskaźnik  do  int   */
   void  * s1;  /* wskaźnik  uniwersalny */

   f1 = &a[3];
   p1 = &a[5];  /* kompilator zgłosi zastrzeżenia */
   s1 = &a[7];
```

Wskaźniki-2

```
    p1 = s1;                /*  kompilator OK   */
    s1 = p1;                /*  kompilator  OK   */
    p1 = (int *) s1;      /*  kompilator OK  */
    p1 = (int *) f1;      /*  kompilator OK  */


/* nie ma sensu   p1 = & 123L;  */
    printf("\n %d   %p   %x\n", *p1,p1,p1);
    printf("\n  %f\n", *f1);


    return (0);
} /*  koniec  funkcji  main  */
```

Wskaźniki-3

1077936128 0xbffffa4c bffffa4c

3.000000

Wskaźniki-4

```
#include <stdio.h>

int main()
{  int   * p1;  /* wskaźnik  do  int   */
   void  * s1;  /* wskaźnik  uniwersalny */


   p1=NULL;


   printf("\n %d   \n", *p1); /* w wykonaniu  segmentation fault */
   printf("\n %p   \n", p1);
   return (0);
} /*  koniec  funkcji  main  */
```

wynik programu:

Segmentation fault (core dumped)

...jeśli tą linię usuniemy to program wypisze

(nil)

Wskaźniki-6

```
#include <stdio.h>
int  main()
{   int   * p1;  /* wskaźnik  do  int   */
    void  * s1;  /* wskaźnik  uniwersalny */

    p1 = (int *) 1;
    printf("\n %d   \n", *p1);  /* segmentation fault */
    printf("\n %p   \n", p1);
    return (0);
} /*  koniec  funkcji  main  */
```

0x1

Wskaźnik-7

```
#include <stdio.h>

int main()

{  float x;

   float  * f2;     /* wskaźnik  do zmiennej typu float */


   f2  = & x ;     /*   !!!! wskaźnik wskazuje na coś rozsądnego  */

   *f2  = 23.4;

    printf("\n wartość x wynosi %5.2f\n",x);

   return (0);

} /*  koniec  funkcji  main  */
```

wynik : 23.40

Wskaźniki-8

(wskaźniki jako argument funkcji call.c str. 213)

```
#include <stdio.h>  /*  autor Steve Oualline  */

void inc_count(int *count_ptr)

{  ++(*count_ptr);      /*   po co nawiasy zwykłe ?   */

}    /*     koniec  funkcji  inc_count     */

     int main ( )

{    int count=0;      /*     licznik pętli     */

     while(count<10)   inc_count(&count);

     printf("\n count=%d\n",count);

}/* koniec      funkcji      main  */
```

Wskaźniki-8 (dygresja)

```
#include <stdio.h>
#include <stdlib.h>
  int main()
  {
   int alfa=10;
   int * wsk;
```

Wskaźniki-8 (dygresja)

```
wsk=&alfa;
    ++(*wsk);
    printf("\n alfa=%d  wsk=%p",alfa,wsk);/* 11    0xbffff2d4 */
    ++*wsk;
    printf("\n alfa=%d  wsk=%p",alfa,wsk);/* 12   0xbffff2d4 */
    (*wsk)++;
    printf("\n alfa=%d  wsk=%p",alfa,wsk);/* 13   0xbffff2d4 */
    *wsk++;
    printf("\n alfa=%d  wsk=%p",alfa,wsk); );/* 13   0xbffff2d8 */
  exit(0);/* dobrze, że koniec, bowiem na co wskazuje wsk ?*/
 } /*  koniec  funkcji  main  */
```

Dygresja

```
#include <stdio.h>
#include <stdlib.h>
  int main()
  {
   int a=11,b=21,c;

   c=a+++b++;/* dla pewności lepiej użyć nawiasów ! */
   printf(" a,b,c   %d   %d   %d\n",  a,  b,  c);
      /*  wypisze   12   22   32  */
   exit(0);
  } /*  koniec  funkcji  main  */
```

Wskaźniki-9
(wskaźniki a macierze)

```
float as[10];     /*   każdy element ma długość 4 bajtów */
float *  p3;


p3 = & as[0];          /*  lub  */          p3=as;


*p3=7;                                  as[0]=7;
*(p3+1)=8;                              as[1]=8;
```

Wskaźniki-10
(wskaźniki a macierze)

```
float as[10];     /*   każdy element ma długość 4 bajtów */

float *  p3;


p3=as;

p3+=4;          /*  p3 = p3 +4      */

*p3=9;          /*  as[4]=9;        */
```

/ dodawanie liczby całkowitej do wskaźnika skaluje się automatycznie */*

Wskaźniki-10 (dygresja)

```
#include <stdio.h>

#include <stdlib.h>

 int main()       /*czy nazwa macierzy jest pełnoprawnym
                   wskaźnikiem?*/

{   int nowa[5]={10,11,12,13,14}; /* inicjalizowanie macierzy */

    int * wsk;  wsk = nowa;

    printf("%d  %d\n", *wsk, *nowa); /*  10   10 */

    printf("%d  %d\n", *(wsk+1), *(nowa+1));/* 11   11 */

    wsk++; /* wskaźnik wsk wskazuje na następną zmienną typu int*/

    /*  nowa++;   tego kompilator nie zaakceptuje...*/        exit(0);

}/*   koniec  funkcji  main  */
```

Wskaźniki-11

```
#include <stdio.h>
    int main ( )
{ float x;
   void   * p2;
   float  * p4;
        x=14;
        p2=&x;
        p4=&x;
            /*    printf("\n %f\n",*p2);      !!??   błędnie!  */
        printf("\n %f\n",*(float *)p2);
        printf("\n %f\n",*p4);
}/* koniec funkcji  main  */
```

Wskaźniki-12

```
int  main()
    char bufa[100]=”Uniwersytet”;
    char bufb[120];
        ........
    copy_string(bufb,bufa);
    copy_string(&bufb[0],&bufa[0]);
.....
  /*     koniec  funkcji  main  */
```

Wskaźniki-13
(str. 220)

```
void  copy_string (char *p,  char  *q)
{
        while (  *p++  =   *q++ ) ;
}  /*    koniec  funkcji  copy_string   */
```

```
    /*   while(   *(p++)  =  *(q++)  )   ;  czytelniejsze!   */
```

Wskaźniki
(wskaźnik do funkcji)

```
#include <stdio.h>

#include <stdlib.h>

     int f1(float);

     int f2(float);

     void zz ( int(*aa)(float), float t );

     int (*f)(float);  /*   f jest wskaźnikiem do funkcji !  */

 int  main( )

{  int n;     scanf("%d",&n);
```

Wskaźniki
(wskaźnik do funkcji)

```
 if(n==1)
    { f=f1;  zz(f, (float)n ); }
  else
         if(n==2)
         {f=f2;  zz(f, (float)n ); }


exit(0);
}   /*   koniec   funkcji   main     */
```

Wskaźniki
(wskaźnik do funkcji)

```
int f1(float x)
{printf("\n wewnatrz f1    x=%f\n",x); return(0);
}/*   koniec  funkcji  f1  */
   int f2(float x)
   {  printf("\n wewnatrz f2    x=%f\n",x); return(0);
   }/*  koniec  funkcji  f2  */
 void zz ( int(*aa)(float), float t )
 { (*aa)(t);
 } /*   koniec   funkcji    zz  */
```

Wskaźnik powinien na coś wskazywać (choćby na NULL)

```
float  * f4 = NULL;


/*  trzeba uważać by obiekt wskazywany nie zniknął !!
*/
```

macierz dwuwymiarowa

```
#include <stdio.h>
#include <stdlib.h>
int a[3][3];
int n,k;
int *wsk;
int main()
{ for(n=0;n<3;++n)
  for(k=0;k<3;++k)
  a[n][k]=10*n+k;
  wsk= (int*) a;
  for(n=0;n<9;++n)
  { printf("\n *wsk=%d",*wsk); wsk++;}
  printf("\n");/* wniosek: macierz jest przechowywana w pamieci wierszami  */
}/* koniec main */
```

macierz dwuwymiarowa

*wsk=0

*wsk=1

*wsk=2

*wsk=10

*wsk=11

*wsk=12

*wsk=20

*wsk=21

*wsk=22

.....powyżej wynik wykonania programu.....

pointer.c

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

/*   prototyp  */
char *dec_bin (unsigned long a);

int main ()
{

  char *bin_p;

  bin_p = (dec_bin (119L));  /* 119 to dziesietnie znak  w */

  printf ("\nTeraz w main: %p\n", bin_p);

  printf (" Czyli:%c\n\n", *(bin_p));
   exit(0);
}       /*  koniec funkcji main */


char * dec_bin (unsigned long dec)
{
  char *wsk;
  char litera_b;
  litera_b=dec;

  printf ("\n wewnatrz funkcji: dec=%ld\n", dec);

  wsk = &litera_b;

  printf ("\nwewnatrz funkcji: %p\n", wsk);
  printf (" Czyli:%c\n\n", *wsk);
  return (wsk);
}       /*   koniec funkcji  dec_bin  */
```

operatory bitowe - przypomnienie

umożliwiają korzystanie, wykonywanie działań na pojedynczych bitach; zaś np. operator dodawania działa na całej zmiennej!

& koniunkcja bitowa

| alternatywa bitowa

^ różnica symetryczna

~ negacja bitowa

<< operator przesunięcia w lewo

>> operator przesunięcia w prawo