Język ANSI C

część 2

Podstawy języka C

Jarosław Gramacki
Instytut Informatyki i Elektroniki

The C Library Reference

```
1.   Language
     1.1  Characters
          1.1.1     Trigraph Characters
          1.1.2     Escape Sequences
          1.1.3     Comments
     1.2  Identifiers
          1.2.1     Keywords
          1.2.2     Variables
          1.2.3     Enumerated Tags
          1.2.4     Arrays
          1.2.5     Structures and Unions
          1.2.6     Constants
          1.2.7     Strings
          1.2.8     sizeof Keyword
     1.3  Functions
          1.3.1     Definition
          1.3.2     Program Startup
     1.4  References
          1.4.1     Pointers and the Address Operator
          1.4.2     Typecasting
     1.5  Operators
          1.5.1     Postfix
          1.5.2     Unary and Prefix
          1.5.3     Normal
          1.5.4     Boolean
          1.5.5     Assignment
          1.5.6     Precedence
     1.6  Statements
          1.6.1     if
          1.6.2     switch
          1.6.3     while
          1.6.4     do
          1.6.5     for
          1.6.6     goto
          1.6.7     continue
          1.6.8     break
          1.6.9     return
     1.7  Preprocessing Directives
          1.7.1     #if, #elif, #else, #endif
          1.7.2     #define, #undef, #ifdef, #ifndef
          1.7.3     #include
          1.7.4     #line
          1.7.5     #error
          1.7.6     #pragma
          1.7.7     Predefined Macros
```

```
2.   Library
     2.1  assert.h
     2.2  ctype.h
     2.3  errno.h
     2.4  float.h
     2.5  limits.h
     2.6  locale.h
     2.7  math.h
     2.8  setjmp.h
     2.9  signal.h
     2.10 stdarg.h
     2.11 stddef.h
     2.12 stdio.h
     2.13 stdlib.h
     2.14 string.h
          2.14.1    Variables and Definitions
          2.14.2    memchr
          2.14.3    memcmp
          2.14.4    memcpy
          2.14.5    memmove
          2.14.6    memset
          2.14.7    strcat
          2.14.8    strncat
          2.14.9    strchr
          2.14.10   strcmp
          2.14.11   strncmp
          2.14.12   strcoll
          2.14.13   strcpy
          2.14.14   strncpy
          2.14.15   strcspn
          2.14.16   strerror
          2.14.17   strlen
          2.14.18   strpbrk
          2.14.19   strrchr
          2.14.20   strspn
          2.14.21   strstr
          2.14.22   strtok
          2.14.23   strxfrm
     2.15 time.h
```

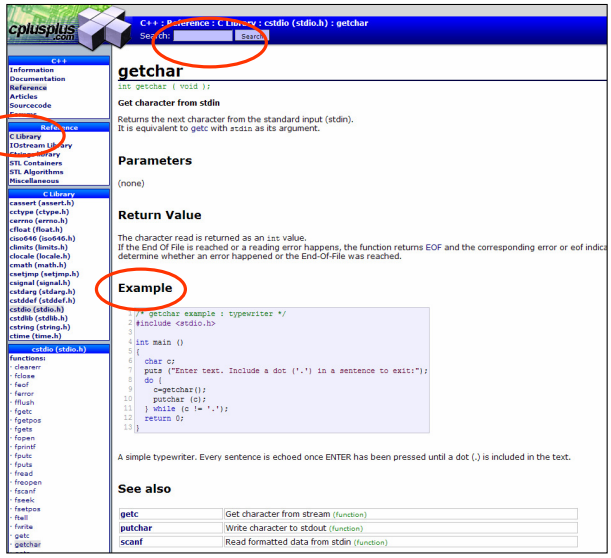
Tylko nazwy nagłówków

Tylko string.h pokazano dokładnie

http://www.acm.uiuc.edu/webmonkeys/book/c_guide/

The C Library Reference

- Wygodny podręczny materiał referencyjny C oraz C++
  - http://www.cplusplus.com/reference/

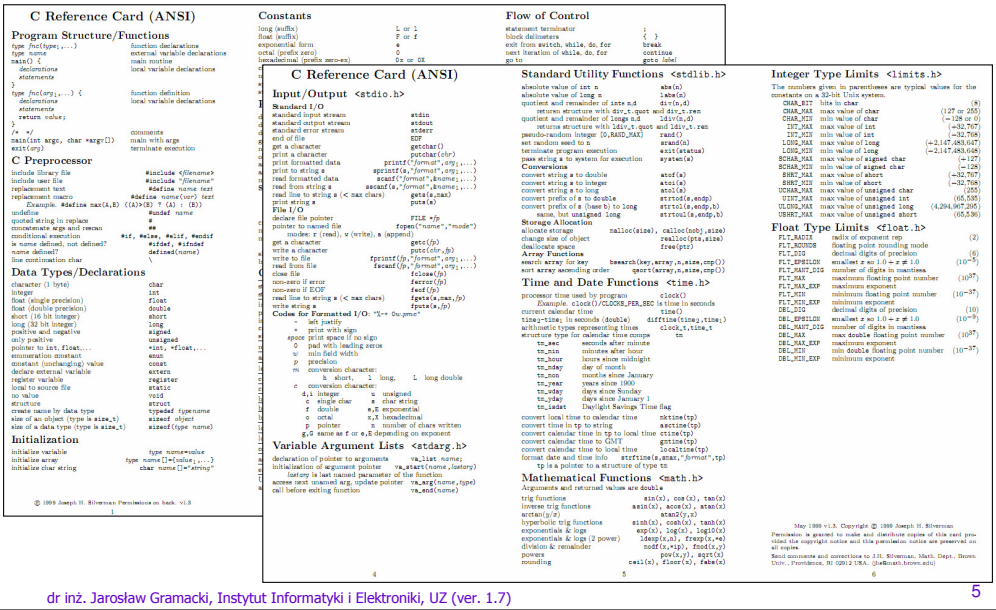


The C Library Reference

- Inna strona
  - http://www.cppreference.com/wiki/c/start
  - Wybierz stronę, która Tobie odpowiada i ... stosuj

The C Library Reference

- Miej zawsze pod ręką ...



C Reference Card (ANSI)

Program Structure/Functions

C Preprocessor

Data Types/Declarations

Initialization

Constants

Flow of Control

C Reference Card (ANSI)

Input/Output <stdio.h>

Variable Argument Lists <stdarg.h>

Standard Utility Functions <stdlib.h>

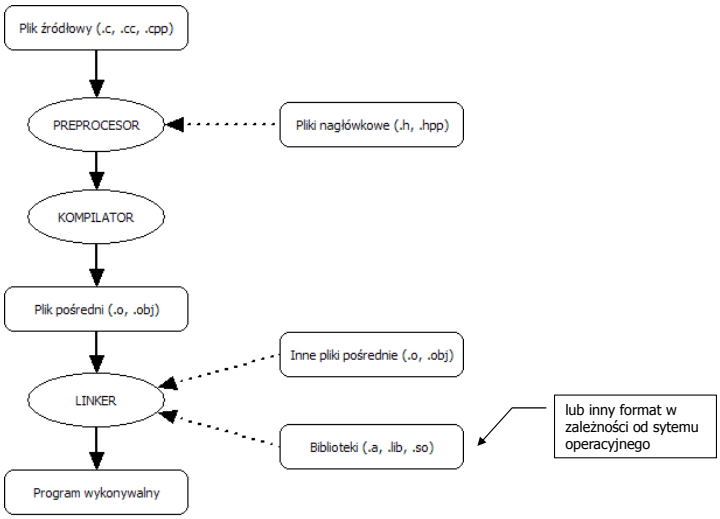
Time and Date Functions <time.h>

Mathematical Functions <math.h>

Integer Type Limits <limits.h>

Float Type Limits <float.h>

Kompilacja programu



- korzystając z IDE (np. Eclipse CDT) całość wykonuje się „jednym kliknięciem”

Cechy języka C
(wybrane w przypadkowej kolejności)

- duże i małe litery są rozróżniane !
- w C bardzo dużo funkcji bibliotecznych
  - wg. wymagań ANSI
- brak instrukcji wejścia-wyjścia w języku
  - (operacje we/wy realizowane przez funkcje biblioteczne)
- brak procedur (tylko funkcje) + funkcji nie można w sobie zagłębiać
- brak typu logicznego (Pascal: `boolen`), napisowego (Pascal: `string`), zbiorowego (Pascal: `set of`)
  - brak operacji (operatorów) na napisach, tablicach, zbiorach
- inny typ char (Pascal: znaki ASCI, C: liczby)
- każde wyrażenie ma swoje wartość liczbową
  - bardzo wygodne rozwiązanie
  - `'a' - 'A', 3 + ( a<1 ), Tab[ a<1 ]`

Cechy języka C
(wybrane w przypadkowej kolejności)

- tablice indeksuje się od 0 !
- jednoindeksowe tablice
  - `A[2][4]` - 1-wymiarowa tablica tablic 4-elementowych)
  - `double A Tab[10][20], *p, **q;`
    `p = Tab[3]; q = Tab;`

wskaźniki, o tym będzie jeszcze mowa

- wskaźniki są wszędzie !!!
- w C argumenty funkcji zawsze są przekazywane przez wartość
  - w C++ dochodzą referencje
  - w Pascalu przez wartość lub przez zmienną

ta uwaga będzie rozwinięta przy omawianiu wskaźników i w szczególności tablic

- wywołanie funkcji z użyciem operatora
  - Pascal: `z := fun;` C: `z =fun();`
- funkcja `main` rozpoczyna wykonanie programu
  - Pascal: blok `begin … end.`

Cechy języka C
(wybrane w przypadkowej kolejności)

- C jest bardzo zwięzły
  - Pascal jest (dla początkujących) bardziej czytelny
  - język C jest dużo bardziej elastyczny niż Pascal. Wyrażenia w nim budowane mogą więc być bardzo złożone (i trudne do zrozumienia)
  - `for(i=1, j=5*i+z; (i+j)/2 > 0; i++, j--, z=z+5);`
- w C łatwo manewrować wskaźnikami, dynamicznie przydzielać pamięć
- słaba kontrola przed korzystaniem z niedozwolonych obszarów pamięci
  - w C programista musi uważać na więcej spraw
- w C są unie
- w C wskaźniki do funkcji
- w C jest wygodna, szybka i bardzo elastyczna instrukcja `for`

Cechy języka C
(wybrane w przypadkowej kolejności)

- International Obfuscated C code contest
  - `http://www.de.ioccc.org/main.html`
    - » To write the most Obscure/Obfuscated C program under the rules below.
    - » To show the importance of programming style, in an ironic way.
    - » To stress C compilers with unusual code.
    - » To illustrate some of the subtleties of the C language.
    - » To provide a safe forum for poor C code. :-)

```
main(O){int I,Q,l=O;if(I=l*4){l=6;if(l>5)l+=Q-8?l-(Q=getchar()-
2)%2:l;if(Q*=2)O+="has dirtiest IF"[(I/-Q&12)-
l/Q%4];}printf("%d\n",8+O%4);}
```

do każdego kodu dołączane są instrukcje jego kompilacji i uruchomienia, wynik działania, etc. (makefile, plik hint, …)

Cechy języka C
(wybrane w przypadkowej kolejności)

```
/*                                      ,*/
          #include                                <time.h>
           #include/*                          _  ,o*/  <stdlib.h>
           #define  c(C)/*     _        . */return      (_C); /*    2004*/
            #include   <stdio.h>/*.   Moekan          "'   `\b-'      */
              typedef/*  */char   p;p* u                ,w        [9
                 ][128] ,*v;typedef  int _;_   R,i,N,I,A              ,m,o,e
           [9],  a[256],k    [9], n[              256];FILE*f       ;_ x   (_ K,_ r
      ,_ q){;   for(;                                        r<     q    ; K       =((
   0xffffff)   &(K>>8))^                                       n[255     &        ( K
  ^u[0        +                                    r  ++      ]     )];c          (K
 )}        _ E                          (p*r,     p*q  ){     c(          f          =
        fopen                        (r  ,q))_  B(_ q){c(    fseek         (f,      0
     ,q))}_ D(){c(  fclose(f ))}_  C(  p    *q){c(  0-     puts(q    )    )}_/*   /
    */main(_ t,p**z){if(t<4)c(   C("<in"       "file>"    "\40<1"   "a"  "yout> "
   /*b9213272*/"<outfile>"   ) )u=0;i=I=(E(z[1],"rb")) ?B(2)?0 :   (((o   =ftell
  (f))>=8)?(u     =(p*)malloc(o))?B(0)?0:!fread(u,o,1,f):0:0)?0:  D():0      ;if(
 !u)c(C("      bad\40input  "));if(E(z[2],"rb" )){for(N=-1;256> i;n[i++] =-1   )a[
 i]=0;       for(i=I=0;   i<o&&(R  =fgetc(   f))>-1;i++)++a[R] ?(R==N)?( ++I>7)?(n[
N]+1       )?0:(n [N   ]=i-7):0:   (N=R)    |(I=1):0;A =-1;N=o+1;for(i=33;i<127;i++
)(        n[i    ]+  l&&N>a[i])?    N= a     [A=i]     :0;B(i=I=0);if(A+1)for(N=n[A];
I<       8&&   (R  =fgetc(f ))>    -1&& i   <o         ;i++)(i<N||i>N+7)?(R==A)?((*w[I
]            =u [i])?1:(*w[I]=   46))?(a              [I++]=i):0:0:0;D();}if(I<1)c(C(
           "  bad\40la" "yout  "))for(i              =0;256>(R=  1);n[i++]=R)for(A=8;
          A  >0;A --)   R  = ( (R&1)==0)           ?(unsigned int)R>>(01):((unsigned
         /*kero  Q'        ,KSS  */)R>>          1)^         0xedb88320;m=a[I-1];a[I
         ]=(m          <N)?(m=   N+8):         ++         m;for(i=00;i<I;e[i++]=0){
        v=w          [i]+1;for(R                      =33;127  >R;R++)if(R-47&&R-92
        &&          R-(_)* w[i])*(                  v++)=     (p)R;*v=0;}for(sprintf
             /*'   _  G*/  (*w+1,                "%0"     "8x",x(R=time(i=0),m,o)^~
           0)    ;i<       8;++           i)u      [N+ i]=*(*w+i+1);for(*k=x(~
             0,i=0     ,*a);i>-        1;      ){for (A=i;A<I;A++){u[+a [ A]
           ]=w[A       ][e[A]]  ;     k     [A+1]=x (k[A],a[A],a[A+1]
         );}if     (R==k[I])      c(      (E(z[3  ],"wb+"))?fwrite(
        /* */   u,o,1,f)?D          ()|C(" \n    OK."):0    :C(
       "  \n  WriteError"            ))  for  (i  =+I-
       1  ;i >-1?!w[i[++               e[+ i]]:0;
         ) for( A=+i--;                 A<I;e[A++]
         =0); (i <I-4                  )?putchar
         ((_    )  46)                 | fflush
        /*'        ,*/                  ( stdout
        ):        0&                    0;}c(C
        ("       \n                    fail")
        )       /*                     dP' /
               dP                      pd  '
              '                        zc
                                      */
                                     }
```


Cechy języka C
(wybrane w przypadkowej kolejności)

```
#include  <stdio.h>
#define   l "================================================="
#define   m "                                                 "
int       _(
float     x,
float     y){
return    x>=0&&x<50&&y>=0&&y<20;}
int main  (){
char      a[2][20][51],c[9][4][51][51],d[100];
int       e,f,g,h=0,i,j=0,k,b=0,n[9],o[9],p[9],x[9],y[9],q[9],r,s,t;
float     u[9],v[9],w[9],z[9];
printf    ("i%s\033Ypppppppppppppppppp1G0",l);
do sprintf(a[0][j],"%s",m,
sprintf   (a[1][j],"%s",l));
while     (++j<20);
scanf     ("%d %d\n",&r,&s);
gets      (d);
do        {
sscanf    (d,"%d %d %f %f %f %f %d %d
%d",&o[b],&p[b],&u[b],&v[b],&w[b],&z[b],&x[b],&y[b],&q[b],j=n[b]=i=0);
do do     c[b][i][j][x[b]]=!
gets      (c[b][i][j]);
while     (++j<y[b]);
while     (++i<q[b]+(j=0)||!++b);}
while     (
gets      (d)||(i=t=e=f=0));
do        {
do strncpy(a[h][i],m,50);
while     (++i<20||(i=0));
do if     (t>=o[i]&&t<=p[i]&&!(k=j=0))
do do if  (_(u[i]+k,v[i]+j))a[h][(
int       )v[i]+j][(
int       )u[i]+k]=c[i][n[i]][j][k];
while     (++k<x[i]);
while     (++j<y[i]+(k=0)||((++n[i]-q[i]||(n[i]=0)),u[i]+=w[i],v[i]+=z[i],0));
while     (++i<b||(k=j=g=0));
do do if  (a[1-h][j][k]!=a[h][j][k]){f-j&&
printf    ("%dG0",j+1,e=0,g+=4);k&&e>k&&
printf    ("0%d ",k,g+=4);k&&e<k&&k-e>1&&
printf    ("%d ",k-e,g+=3);k&&e<k&&k-e==1&&
printf    (" ",g++);
printf    ("r%c",a[h][j][k],g+=2,e=k,f=j);}
while     (++k<50);
while     (++j<20+(k=0));
do printf ("1G0",g+=3,e=f=0);
while     (g<s);h=1-h;}
while     (++t<=r+(i=0));
return    !
puts      ("");}
```

Cechy języka C
(wybrane w przypadkowej kolejności)

```
#include <ncurses.h>/*****************************************************/
        int             m[256                ] [         256   ],a
 ,b   ;;;   ;;;   WINDOW*w;   char*l=""   "\176qxl"   "q"   "q"   "k"   "w\
xm"   "x"   "t"         "j"         "v"         "u"         "n"         ,Q[
 ]=   "Z"   "pt!ftd`"   "qdc!`eu"   "dq!$c!nnwf"/**   ***   */"t\040\t";c(
int   u ,         int         v){                 v?m   [u]         [v-
 1]   |=2,m[u][v-1] &   48?W][v-1   ] &   15]]):0:0;u?m[u   -1][v]|=1   ,m[
 u-          l][   v]&         48?               W-1   ][v         ]&
15]   ]):0:0;v<   255   ?m[   u][v+1]|=8,m[u][v+1]&   48?   W][   v+1]&15]]
):0         :0;         u <         255   ?m[   u+1         ][v   ]|=
4,m[u+1][   v]&48?W+1][v]&15]]):0:0;W][   v]&   15]   ]);}cu(char*q){   return
 *q                 ?cu   (q+         1)&         1?q   [0]                 ++:
q[0   ]--   :1;   }d(   int   u ,   int/**/v,   int/**/x,   int   y){   int
Y=y   -v,   X=x         -u;   int         S,s   ;Y<         0?Y   =-Y   ,s,
s=-   1:(   s=1);X<0?X=-X,S   =-1   :(S=   1);   Y<<=   1;X<<=1;   if(X>Y){
int   f=Y         -(X   >>1   );;                 while(u!=         x){
f>=   0?v+=s,f-=X:0;u   +=S   ;f+=   Y;m[u][v]|=32;mvwaddch(w,v   ,u,   m[u
 ][         v]&   64?   60:         46)         ;if         (m[   u][
v]&16){c(u,v);;   ;;;   ;;;   return;}}   }else{int   f=X   -(Y>>1);;   while
 (v   !=y         ){f   >=0         ?u   +=S,                 f-=         Y:0
 ;v   +=s   ;f+=X;m[u][v]|=   32;mvwaddch(w,v   ,u,m[u][v]&64?60:46);if(m[u
 ][                 v]&         16)   {c(   u,v                           );
  ;   return;;;}}}}Z(   int/**/a,   int   b){   }e(   int/**/y,int/**/   x){
int         i ;         for         (i=         a;i         <=a
+S;i++)d(y,x,i,b),d(y,x,i,b+L);for(i=b;i<=b+L;i++)d(y,x,a,i),d(y,x,a+   S,i
 );         ;;;         ;;;         ;;;         ;;;         ;;;   ;
 mvwaddch(w,x,y,64);   ;;;   ;;;   ;;;   prefresh(   w,b,a,0,0   ,L-   1,S-1
);}         main(         int                 V ,   char                *C[
  ]   ){FILE*f=   fopen(V==1?"arachnid.c"/**/   :C[   1],"r");int/**/x,y,c,
v=0         ;;;      initscr             ();                 Z(Z         (raw
 ()   ,Z(   curs_set(0),Z(1   ,noecho()))),keypad(    stdscr,TRUE));w   =newpad
  (   300,  300                 ) ;   for         (x=   255   ; x   >=0   ;x--
  )   for   (y=   255   ;y>=0;y--   )m[   x][   y]=   0;x=y=0;refresh( );while
  (   (c=                 fgetc (f)   )+1)                  {if(
0||c==10||  x==   256){x=0;y++;if(y==256  )break;;}   else{m[x][y]=(c   ==
'~'   ?64   : c   ==32         ?0:         16)   ;;x                  ++;
      }}for(x=0   ;x<   256;x++)m   [x][0]=16   ,m[   x][   255]=16;for(y=0
;y<         256   ; y         ++)   m[0         ][y   ] =                 16,
m[255][y]   =16   ;a=b=c=0;   x=y   =1;   do{v++;mvwaddch   (w,   y,x   ,m[
x][         y]&         32?   m[x                 ][y   ] &   16?
 0|   acs_map[l[m[x][y]&15]]:46 :   32);c==0163&&!(m[x][y+1]&16)?y++:   0;c
 ==   119         &&!         (m[                 x][
 y-   1]&   16)   ?y--:0;;c   ==97   &&!(m[x-1][y]&16)?x--:0;c==100&&!(m[x+1
 ][   y]&   16)         ? x   ++:0                ;if(                      c==
 3-   1+1   )(endwin(   );;   return(0)   ;}x   -a<5?a>S-   5?a-=S-5:(a=0):
0;x                 -a>         S-5?a<255   -S*         2?a               +=S
-5:(a=256-S):0;   y-b<5?b>L-5?b-=L-5:(b   =0)   :0;   y-b>L-5?b<255-L   *2?
b+=                         L-5   :(b                 =256
-L)   :0;e(x,y);if(m[x][y]&64)break;}while((c=getch())!=-1);endwin();cu(Q);
printf(Q,v);}
```


Biblioteka standardowa

- Język C, w przeciwieństwie do innych języków programowania (np. Fortranu czy Pascala) nie posiada absolutnie **żadnych** słów kluczowych, które odpowiedzialne by były za obsługę wejścia i wyjścia
  - » brak podstawowych funkcji wejścia-wyjścia jest jednak jedną z największych zalet tego języka
  - » m.in. właśnie dzięki temu programy napisane w języku C są bardzo szybkie
- W 1983 roku, kiedy zapoczątkowano prace nad standaryzacją C, zdecydowano, że powinien być zestaw instrukcji identycznych w każdej implementacji C. Nazwano je **Biblioteką Standardową** (czasem nazywaną "**libc**")
- Zawiera ona podstawowe funkcje, które umożliwiają wykonywanie takich zadań jak
  - » wczytywanie i zwracanie danych <stdio.h>
  - » modyfikowanie zmiennych łańcuchowych <string.h>
  - » działania matematyczne <math.h>
  - » operacje na plikach <stdio.h>
- libc nie zawiera żadnych funkcji, które mogą być zależne od systemu operacyjnego czy sprzętu, jak grafika, dźwięk czy obsługa sieci

```
ANSI Standard Libraries
<assert.h> <ctype.h> <errno.h>  <float.h>  <limits.h>
<locale.h> <math.h>  <setjmp.h> <signal.h> <stdarg.h>
<stddef.h> <stdio.h> <stdlib.h> <string.h> <time.h>
```

"Wady" języka C
(tylko kilka wybranych)

- C był tworzony jako mały i prosty język, co niewątpliwie przyczyniło się do jego popularności
  - wynikają z tego jednak „wady", co powoduje, że wiele prostych błędów programistycznych nie jest wykrywanych przez kompilator, a przy wykonywaniu programu ujawniają się dopiero po jakimś czasie
  - wielka zwięzłość wyrażeń
    - » zbyt wiele w jednej linii kodu C !
  - brak sprawdzania zakresu tablic
    - » bardzo niebezpieczne tzw. błędy przepełnienia bufora
  - brak funkcji zagnieżdżonych
  - brak obsługi wyjątków
  - brak standardowych bibliotek graficznych
  - wiele operacji w C mających niezdefiniowane zachowanie nie jest sprawdzanych w czasie kompilacji
    - » odczyt i zapis poza zasięgiem tablicy
    - » przekroczenie zakresu liczb całkowitych
    - » odczytanie zmiennej przed zapisaniem do niej wartości
    - » kolejność wykonywania wyrażeń przekazanych jako argumenty do funkcji
  - ... i naprawdę wiele innych !

Struktura programu

```
int fun (int, int, int c);          // prototyp funkcji, deklaracja

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

#define MAX 5                       // prosty przypadek użycia #define

int x = 2, y = -7, z;               // zm. globalne

int fun (int a, int b, int c)       // definicja funkcji
  {
  int lokalna = 10;                 /* zm. lokalna */
  return (a + b - c + lokalna);
  }

int main (void)
  {
  int wynik;

  x = x + y * z - MAX;              // ile wynosi z ?
  wynik = fun (x, abs(y), y+2);     // abs() - f. standardowa z stdlib.h
  printf ("Wynik działania: %d", wynik);

  return 0;                         // program poprawnie zakończony
  }                                 // lub EXIT_SUCCESS
```

standardowe pliki nagłówkowe - K&R Dodatek B

BARDZO WAŻNE formatowanie kodu **spacje !!!**

Funkcje
(tylko krótki wstęp)

- w wielu podręcznikach funkcje wprowadzane są dosyć późno.
- z funkcji należy jednak korzystać nawet w prostych programach (nawyk !)
- zbyt dużo programów ma formę
  - `void main(void) { ... kilka stron kodu ... }`

Pascal:

```
function fun ( a, b, c : integer) : integer
  begin
    fun := a + b - c;
  end;   { separator instrukcji }
```

C:

```
int fun (int a, int b, int c)      // arg. tego samego typu
                                   // nie łączymy, brak średnika na końcu
  {
  return a + b - c;
  // return (a + b - c);           // też OK
  }

dummy() {}                         // nic nie robi, zwraca int
void dummy() {}                    // nic nie robi, nic nie zwraca
```


Typy danych, zmienne

`void`	typ pusty (wykorzystywany głównie w operacjach na wskaźnikach i dla wskazania braku parametrów i/lub wyniku)
całkowite:	
`char`	1 bajt
`int`	2 lub 4 bajty (zależy to od platformy)
`long (long int)`	4 lub 8 bajtów (zależy to od platformy)
`enum`	2(4) bajty (zbiór wartości) // typ wyliczeniowy
	`enum boolean {FALSE, TRUE}; boolean b;`
zmiennoprzecinkowe:	
`float`	4 bajty dokładność ok. 7 cyfr znaczących
`double`	8 bajtów dokładność ok. 15 cyfr znaczących
`long double`	8,10,12 lub 16 bajtów
kwalifikatory zmiennych:	
`short, long`	np. short int, long int, long double
`signed, unsigned`	np. signed int, unsigned int
	(od maszyny zależy, który jest domyślny)

Integer Type Limits:	<limits.h>
Float Type Limits:	<float.h>

Typy danych, zmienne

- reprezentacja wewnętrzna:
  - dla zmiennych całkowitych: kod U2 lub NKB (naturalny kod binarny)
  - dla zmiennych zmiennoprzecinkowych:
    - » standard IEEE 754 Standard

```
Exponent 8      Mantissa 23+1
S EEEEEEEE      FFFFFFFFFFFFFFFFFFFFFFF
0 1       8     9                     31
```

O liczbach zmiennoprzecinkowych, ich reprezentacjach wewnętrznych oraz związanymi z nimi zagadnieniami błędów obliczeń będzie mowa dokładniej na oddzielnym wykładzie

Typy danych, zmienne

(dla uproszczenia operujemy na liczbach 4. bitowych)

kod NKB:
decyduje suma wag bitów

```
1001(2)           = 9(10)

1     0     0     1
2^3   2^2   2^1   2^0
1*8 + 0*4 + 0*2 + 1*1
```

kod U2:
decyduje suma wag bitów oraz znak liczby

```
0111(2)           = 7(10)
1001(2)           = -7(10)

 1     0     0     1
-2^3   2^2   2^1   2^0
-1*8 + 0*4 + 0*2 + 1*1    = -7(10)
```

"szybka" metoda:
-a = ~a + 1
znajdź postać BIN U2 liczby -7

```
25   = 00011001
~25  = 11100110
              1 +
       ---------
       11100111 = -25
```

zwiększenie rozmiaru liczby (na przykład przy konwersjach):
dla liczby BEZ znaku - przez uzupełnienie zerami

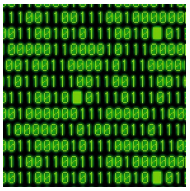
```
1001(2)           = 9(10)
00001001(2)       = 9(10)
```

dla liczby ZE znakiem - przez powielenie bitu znaku

```
1001(2)           = -7(10)
11111001(2)       = -7(10)
```

Typy danych, zmienne

zapis BIN a zapis HEX:



- liczby DEC "nie nadają się" dla informatyka i komputera
- liczby BIN są za długie
- liczby HEX bardzo skracają zapis

- szalenie wygodny "przelicznik" z HEX na BIN
- każda cyfra liczby HEX odpowiada czterem bitom liczby BIN

0xAB1F(16) 1010 1011 0001 1111(2)
A B 1 F
wagi: 8421 8421 8421 8421
zamiast: 2^15 ... 64 32 16 8 4 2 1

- Zapisać (na 16 bitach) w postaci bin i hex liczbę 5000

- Bin:
0001 0011 1000 1000 (2)
2^12 + 2^9 + 2^8 + 2^7 + 2^3 = 5000 (10)

- Hex:
0x1388 (16)

Typy danych, zmienne

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

int main(void)
{
  int i, d = 5000;
  float w;

  for(i=0; i<16; i++)
    {
    w = (float)d / 2;
    printf("%5d %7.2f %4d 2\^%d\n", d, w, d%2, i);
    d = (int)w;
    }
  return 0;
}
```

```
0001 0011 1000 1000 (2)
2^12 + 2^9 + 2^8 + 2^7 + 2^3 = 5000 (10)
```

```
 5000 2500.00    0 2^0
 2500 1250.00    0 2^1
 1250  625.00    0 2^2
  625  312.50    1 2^3
  312  156.00    0 2^4
  156   78.00    0 2^5
   78   39.00    0 2^6
   39   19.50    1 2^7
   19    9.50    1 2^8
    9    4.50    1 2^9
    4    2.00    0 2^10
    2    1.00    0 2^11
    1    0.50    1 2^12
    0    0.00    0 2^13
    0    0.00    0 2^14
    0    0.00    0 2^15
```

Typy danych, zmienne

- uwaga na liczby ze znakiem i bez znaku !
  - postać bitowa kilku wybranych wartości zmiennej typu `char` i `unsigned char` oraz wynik zwracany przez funkcję `printf`

deklaracja: char num //-128 do 127

```
Enter number: 1
Binary of 1 is:     0 0 0 0 -- 0 0 0 1

Enter number: 15
Binary of 15 is:    0 0 0 0 -- 1 1 1 1

Enter number: 127
Binary of 127 is:   0 1 1 1 -- 1 1 1 1

Enter number: 128
Binary of -128 is:  1 0 0 0 -- 0 0 0 0

Enter number: -15
Binary of -15 is:   1 1 1 1 -- 0 0 0 1

Enter number: -127
Binary of -127 is:  1 0 0 0 -- 0 0 0 1

Enter number: -128
Binary of -128 is:  1 0 0 0 -- 0 0 0 0

Enter number: -129
Binary of 127 is:   0 1 1 1 -- 1 1 1 1
```

deklaracja: unsigned char num //0 do 255

```
Enter number: -1
Binary of 255 is:  1 1 1 1 -- 1 1 1 1

Enter number: -2
Binary of 254 is:  1 1 1 1 -- 1 1 1 0

Enter number: -15
Binary of 241 is:  1 1 1 1 -- 0 0 0 1

Enter number: 127
Binary of 127 is:  0 1 1 1 -- 1 1 1 1

Enter number: 128
Binary of 128 is:  1 0 0 0 -- 0 0 0 0

Enter number: 129
Binary of 129 is:  1 0 0 0 -- 0 0 0 1

Enter number: 255
Binary of 255 is:  1 1 1 1 -- 1 1 1 1

Enter number: 256
Binary of 0   is:  0 0 0 0 -- 0 0 0 0

Enter number: 257
Binary of 1 is:    0 0 0 0 -- 0 0 0 1
```


Typy danych, zmienne

- kod programu, który posłużył do przygotowania wydruku z poprzedniej strony

```
#include <stdio.h>
#include <limits.h>                // for CHAR_BIT
void main(void) {
  unsigned char i, bit;
  char num = 1;
  unsigned char mask;              // mask MUST be unsigned

  while(num != 0)
  {
    mask = 0x80;                   // 1000 0000 binary
    printf("\nEnter number: ");
    scanf("%d", &num);
    printf("Binary of %d is:  ", num);
    for(i=0; i < CHAR_BIT; i++)    // for each bit
      {
      bit = (mask & num) ? 1 : 0;  // bit is 1 or 0
      printf("%d ", bit);          // print bit
      if(i == 3) printf("-- ");    // print dash between bytes
      mask >>= 1;                  // shift mask to the right
      }
  }
}
```

Operatory bitowe można stosować jedynie do argumentów całkowitych !

Typy danych, zmienne

- trochę bardziej złożony problem: wyświetl te liczby z podanego zakresu, które mają ustawione obok siebie (tu:) sześć bitów

```
#include <stdio.h>
int main(void)
{
  unsigned int maska;
  int i, j, ile=0, zakres;

  scanf("%d", &zakres);
  for(i=1; i<zakres; i++)
  {
    maska = 0xfc000000;  //6 jedynek obok siebie
    for(j=0; j<=26; j++) // 32-6=26
    {
      if ((i & maska) == maska)
        {
          printf("%d\n", i); ile++;
          break;
        }
      maska >>= 1;
    }
  }
 printf("Suma: %d\n", ile);
return 0;
}
```

zastanów się nad sposobem określania w trakcie działania programu ile jedynek ma być koło siebie

```
... ./a.out
456

63
126
127
191
252
253
254
255
319
382
383
447
Suma: 12
```


Stałe

```
znakowe:
literały          'A', 'a', '1', '7'
kody symboliczne  '\101', '\x4F', '\10', '\0'      // 8-kowo lub 16-kowo
opis symboliczny  '\n', '\b', '\\', '\'', '\"'

całkowite:
dziesiętnie       46, 46L, 46l, 46U, 46u, 56UL
ósemkowo          056, 056L, 056l                  // poprzedzone zerem
szesnastkowo      0x2, 0x20L, 0x2ul, 0x1F5B        // poprzedzone 0x

rzeczywiste:
dzisietnie        46.0, 0.025, 39.2F, 89.6L
wykładniczo       4.6e1, -2.5E-2, 896e-1L          // -2.5 10^-2

łańcuchowe:       "Dowolny napis", "\n tekst \n", ""

wyliczeniowe:     enum MyMonths {JAN = 1, FEB, APR, NOV};
```

10(8) czyli 8(10)

- kody symboliczne: tylko w postaci **HEX** lub **OCT**
- stała rzeczywista: bez przyrostka jest typu **double**. Zakończona F jest typu **float**. Zakończona L jest typu **long double**
- "A" - dwubajtowy tekst zakończony znakiem '\0'. Wartością łańcucha "A" jest wskazanie typu char* na pierwszy znak tego łańcucha
- 'A' - literał o wartości 65. Wartością literału 'A' jest liczba całkowita 65 typu int (char w C++)
- stałe są zawsze BEZ znaku. Napis -3456 nie jest liczba ujemną ale wyrażeniem złożonym z operatora 'minus' i liczby dodatniej