Biblioteki ANSI C

Standard ANSI definiuje wszystkie aspekty C: język, preprocesor oraz bibliotekę. Prototypy funkcji bibliotecznych, jak również niezbędne struktury danych oraz stałych preprocesora są zdefiniowane w standardowych zbiorach nagłówkowych. Poniższa tabela zawiera zestawienie standardowych zbiorów nagłówkowych, włącznie z podsumowaniem ich zawartości.

STANDARDOWE ZBIORY NAGŁÓWKOWE W ANSI C	
Zbiór nagłówkowy	Przeznaczenie
assert.h	Definiuje makro *assert*. Używane do diagnostyki programu.
ctype.h	Deklaruje funkcje do klasyfikacji i konwersji znaków.
errno.h	Definiuje makrosy dla warunków błędu, *EDOM* i *ERANGE*, oraz zmienna całkowitą *errno* poprzez którą funkcje biblioteczne zwracają kod błędu.
float.h	Definiuje zakres wartości, które mogą być zapamiętywane w typach zmiennoprzecinkowych.
limits.h	Definiuje wartości ograniczające dla wszystkich danych typu całkowitego.
locale.h	Deklaruje strukturę *lconv* i funkcje niezbędne do przystosowania programu w C do poszczególnych środowisk lokalnych.
math.h	Deklaruje funkcje matematyczne oraz makro *HUGE_VAL*
setjmp.h	Definiuje funkcje *setjmp* oraz *longjmp*, które mogą przekazywać kontrolę z jednej funkcji do drugiej bez opierania się na wywołaniach i powrotach z funkcji. Definiuje również typ danych *jmp_buf* używany przez *setjmp* i *longjmp*.
signal.h	Definiuje symbole i procedury niezbędne dla obsługi zdarzeń wyjątkowych.
stdarg.h	Definiuje makrosy, które umożliwiają dostęp do nienazwanych argumentów w funkcji, która akceptuje zmienną liczbę argumentów.
stddef.h	Definiuje standardowe typy danych *ptrdiff_t*, *size_t*, *wchar_t*, symbol *NULL* oraz makro *offsetof*
stdio.h	Deklaruje funkcje i typy danych niezbędne do obsługi operacji we/wy. Definiuje makrosy takie jak *BUFSIZ*, *EOF*, *NULL*, *SEEK_CUR*, *SEEK_END* i *SEEK_SET*
stdlib.h	Deklaruje wiele funkcji użyteczności takich, jak procedury konwersji łańcuchów, generator liczb losowych, procedury alokacji pamięci i procedury kontroli procesów (takie, jak *abort*, *exit* i *system*).
string.h	Deklaruje funkcje do manipulowania łańcuchami takie, jak *strcmp* i *strcpy*.
time.h	Deklaruje typy danych i definiuje funkcje do manipulowania czasem. Definiuje typy *clock_t* i *time_t* oraz strukturę danych *tm*

Uwaga: Identyfikatory zewnętrzne rozpoczynające się od znaku podkreślenia są zarezerwowane na użytek biblioteki: to samo dotyczy wszelkich innych identyfikatorów rozpoczynających się od znaku podkreślenia i wielkiej litery lub dwóch znaków podkreślenia.

Wejście i wyjście: nagłówek `<stdio.h>`

Operacje na plikach

```
FILE *fopen(const char *filename, const char *mode)

    FILENAME_MAX maksimum znakow w nazwie pliku

    FOPEN_MAX    plikow mozna jednoczesnie otworzyc

FILE *freopen(const char *filename,
              const char *mode, FILE *stream)
    Zwykle uzywa sie do zmiany plikow zwiazanych
    ze strumieniami stdin, stdout, stderr

#include <stdio.h>

int main(void)
{
   /* redirect standard output to a file */
   if (freopen("OUTPUT.FIL", "w", stdout)
       == NULL)
      fprintf(stderr, "error redirecting stdout\n");

   /* this output will go to a file */
   printf("This will go into a file.");

   /* close the standard output stream */
   fclose(stdout);

   return 0;
}

int fflush(FILE *stream)

int fclose(FILE *stream)

int remove(const char *filename)

int rename(const char *oldname, consat char *newname)

FILE *tmpfile(void)

char *tmpnam(char s[L_tmpnam])

   TMP_MAX  co najwyzej tyle roznych nazw mozna
            wygenerowac podczas dzialania
            jednego programu

int setvbuf(FILE *stream, char *buf,
                int mode, size_t size)
    mode - mozliwe 3 wartosci:
      _IOFBUF  pelne buforowanie
      _IOLBUF  buforowanie wierszy pliku tekstowego
      _IONBUF  brak buforowania
    Jezeli buf rowny NULL, to przydzial domyslny.

#include <stdio.h>
int main(void)
{
   FILE *input, *output;
```

```
   char bufr[512];

   input = fopen("file.in", "r+b");
   output = fopen("file.out", "w");

   /* set up input stream for minimal disk access,
      using our own character buffer */
   if (setvbuf(input, bufr, _IOFBF, 512) != 0)
      printf("failed to set up buffer for input "
             "file\n");
   else
      printf("buffer set up for input file\n");
   /* set up output stream for line buffering
      using space that will be obtained through
      an indirect call to malloc */
   if (setvbuf(output, NULL, _IOLBF, 132) != 0)
      printf("failed to set up buffer for output"
             " file\n");
   else
      printf("buffer set up for output file\n");

   /* perform file I/O here */
   /* close files */
   fclose(input);
   fclose(output);
   return 0;
}

void setbuf(FILE *stream, char *buf)
     Jesli buf == NULL, to buforowanie zostanie
     wylaczone, w przeciwnym przypadku rownowazne

     (void) setvbuf(stream,_IOFBUF, BUFSIZ)

#include <stdio.h>
/* BUFSIZ is defined in stdio.h */
char outbuf[BUFSIZ];

int main(void)
{
   /* attach a buffer to the standard output stream */
   setbuf(stdout, outbuf);

   /* put some characters into the buffer */
   puts("This is a test of buffered output.\n\n");
   puts("This output will go into outbuf\n");
   puts("and won't appear until the buffer\n");
   puts("fills up or we flush the stream.\n");

   /* flush the output buffer */
   fflush(stdout);

   return 0;
}
```

Formatowane wyjście

```
int fprintf(FILE *stream, const char *format, ... )

int printf(const char *format, ... )

int sprintf(char *s, const char *format, ... )

int vprintf(const char *format, va_list arg)

int vfprintf(FILE *stream, const char *format,
             va_list arg)

int vsprintf(char *s, const char *format, va_list arg)
```

Formatowane wejście

```
int fscanf(FILE *stream, const char *format, ... )

int scanf(const char *format, ... )

int sscanf(char *s, const char *format, ... )
```

Funkcje realizujące wejście i wyjście znakowe

```
int fgtec(FILE *stream)

char *fgets(char *s, int n, FILE *stream)

int fputc(int c, FILE *stream)

int fputs(const char *s, FILE *stream)

int getc(FILE *stream)

int getchar(void)

char *gets(char *s)

int putc(int c, FILE *stream)

int putchar(int c)

int puts(const char *s)

int ungetc(int c, FILE * stream)
```

Funkcje realizujące bezpośrednie wejście i wyjście

```
size_t fread(void *ptr, size_t size, size_t nobj,
             FILE *stream)

#include <string.h>
#include <stdio.h>

int main(void)
{
   FILE *stream;
   char msg[] = "this is a test";
   char buf[20];

   if ((stream = fopen("DUMMY.FIL", "w+"))
       == NULL)
   {
      fprintf(stderr, "Cannot open output file.\n");
      return 1;
   }

   /* write some data to the file */
   fwrite(msg, strlen(msg)+1, 1, stream);

   /* seek to the beginning of the file */
   fseek(stream, SEEK_SET, 0);

   /* read the data and display it */
   fread(buf, strlen(msg)+1, 1, stream);
   printf("%s\n", buf);

   fclose(stream);
   return 0;
}

size_t fwrite(const void *ptr, size_t size,
              size_t nobj, FILE *stream)

#include <stdio.h>
struct mystruct
{
  int i;
```

```
  char ch;
};

int main(void)
{
   FILE *stream;
   struct mystruct s;

   if ((stream = fopen("TEST.$$$", "wb")) == NULL)
   /* open file TEST.$$$ */
   {
      fprintf(stderr, "Cannot open output file.\n");
      return 1;
   }
   s.i = 0;
   s.ch = 'A';
   fwrite(&s, sizeof(s), 1, stream);
   /* write struct s to file */
   fclose(stream); /* close file */
   return 0;
}
```

Funkcje wyznaczające pozycję w pliku

```
int fseek(FILE *stream, long offset, int origin)
    Dla plikow binarnych:
       nowa pozycja w miejscu oddalonym o offset
    znakow od origin.
       Wartosci  origin
        SEEK_SET  poczatek pliku
        SEEK_CUR  biezaca pozycja
        SEEK_END koniec pliku

    Dla plikow tekstowych:
       offset musi byc rowny zeru
         lub wartosci zwroconej przez ftell
         (wowczas origin musi byc rowne SEEK_SET)
    Funkcja fseek zwraca wartosc niezerowa w
    przypadku bledu.

long ftell(FILE *stream)
    zwraca wartosc biezacej pozycji dla stream
    lub -1L w przypadku bledu.

void rewind(FILE *stream)
     Wywolanie rewind jest rownowazne:
      fseek(fp,0L,SEEK_SET);
      clearerr(fp);
```

```
 #include <stdio.h>
 int main(void)
 {
    FILE *fp;
    char *newname="test.txt", first;

    fp = fopen(newname,"w+");
    fprintf(fp,"abcdefghijklmnopqrstuvwxyz");
    rewind(fp);
    fscanf(fp,"%c",&first);
    printf("The first character is: %c\n",first);
    fclose(fp);
    remove(newname);
    return 0;
 }
```

```
int fgetpos(FILE *stream, fpos_t *ptr)

int fsetpos(FILE *stream, const fpos_t *ptr)
    ustawia biezaca pozycje w stream wedlug
    wartosci zapamietanej przez fgetpos w
    miejscu wskazywanym przez ptr
    W przypadku bledu zwraca wartosc rozna od
    zera
```

Obsługa błędów

```
void clearerr(FILE *stream)
     Kasuje znaczniki konca pliku i bledu.

int feof(FILE *stream)
     Zwraca wartosc rozna od 0 na koncu pliku.

int ferror(FILE *stream)
     Zwraca wartosc rozna od zera, gdy blad.

void perror(const char *s)
     Wypisuje komunikat zwiazany z numerem errno.
```

Klasyfikowanie znaków: nagłówek `<ctype.h>`

isalnum(c)	prawdą jest albo isalpha(c) albo isdigit(c)
isalpha(c)	prawdą jest isupper(c) albo islower(c)
iscntrl(c)	znak kontrolny
isdigit(c)	cyfra dziesiętna
isgraph(c)	znak drukowalny za wyjątkiem odstępu
islower(c)	mała litera
isprint(c)	znak drukowalny łącznie z odstępem
ispunct(c)	znak drukowalny za wyjątkiem odstępu, liter i cyfr
isspace(c)	odstęp, nowa strona, nowy wiersz, powót karetki, tabulator, pionowy tabulator
isupper(c)	duża litera
isxdigit(c)	cyfra szesnastkowa

Dwie dodatkowe funkcje służą do zmiany wielkości liter:

```
int tolower(int c)    zamienia c na mala litere
int toupper(int c)    zamienia c na duza litere
```

Operacje na tekstach: nagłówek `<string.h>`

```
char *strcpy(char *s, const char *ct)
   Kopiuje ct do s lacznie z '\0'; zwraca s

char *strncpy(char *s, const char *ct, size_t n)
   Kopiuje co najwyzej n znakow;

char *strcat(char *s, const char *ct)
   Dopisuje znaki z ct na koniec s; zwraca s

int strcmp(const char *cs, const char *ct)
   Porownuje znaki.

int strncmp(const char *cs, const char *ct, size_t n)
   Porownuje co najwyzej n znakow.

char *strchr(const char *cs, char c)
   zwraca wskaznik do pierwszego wystapienia c w s
   lub NULL, jesli znak nie wystepuje

char *strrchr(const char *cs, char c)
   zwraca wskaznik do ostatniego wystapienia c w s

size_t strspn(const char *cs, const char *ct)
   zwraca dlugosc przedrostka w tekscie cs
   skladajacego sie ze znakow wystepujacych w ct
```

```
size_t strcspn(const char *cs, const char *ct)
   zwraca dlugosc przedrostka w tekscie cs sklada-
   jacego sie ze znakow nie wystepujacych w ct

char *strpbrk(const char *cs, const char *ct)
   zwraca wskaznik do pierwszego wystapienia w cs
   jakiegos znaku z ct lub NULL, jesli takich nie ma

char *strstr(const char *cs, const char *ct)
   zwraca wskaznik do pierwszego wystapienia tekstu ct
   w cs lub NULL, jesli ct nie wystepuje w cs

size_t strlen(const char *cs)

char *strerror(size_t n)
   zwraca wskaznik do tekstu komunikatu
   odpowiadajacego bledowi o numerze n

char *strtok(char *s, const char *ct)
   wyszukuje w tekscie s ciagi znakow przedzielone
   znakami z ct
```

Funkcje *mem...* służą do operowania na obiektach traktowanych jak tablice znakowe; z założenia mają one działać bardzo sprawnie.

```
void *memcpy(void *s, const void *ct, size_t n)
   Kopiuje n znakow z obiektu ct do s i zwraca s
void *memmove(void *s, const void *ct, size_t n)
   Robi to samo, co memcpy, ale dziala rowniez
   dla obiektow zachodzacych na siebie
int memcmp(const void *cs, const void *ct, size_t n)
   porownuje poczatkowe n znakow w cp i cs;
   zwraca taka sama wartosc jak strcmnp.
void *memchr(const void *cs, char c, size_t n)
   zwraca wskaznik do pierwszego wystapienia c w s
   lub NULL, jesli nie wystepuje.
void *memset(void *s, char c, size_t n)
   wstawia c do poczatkowych n znakow s; zwraca s
```

Funkcje matematyczne: nagłówek `<math.h>`

W poniższym zestawieniu:

x,y - typu *double*

n - *typu int*

Wszystkie funkcje zwracają wartość typu *double*. Wartości kątów dla funkcji trygonometrycznych wyraża się w radianach.

sin(x)	sinus x
cos(x)	cosinus x
tan(x)	tangens x
asin(x)	$sin^{-1}(x)$ w przedziale $[-\pi 2, \pi 2]$, $x \in [-1, 1]$
acos(x)	$cos^{-1}(x)$ w przedziale $[0, \pi]$, $x \in [-1, 1]$
atan(x)	$tan^{-1}(x)$ w przedziale $[-\pi 2, \pi 2]$
atan2(y,x)	$tan^{-1}(yx)$ w przedziale $[-\pi, \pi]$
sinh(x)	sinus hiperboliczny x
cosh(x)	cosinus hiperboliczny x
tanh(x)	tangens hiperboliczny x
exp(x)	funkcja wykładnicza e^x
log(x)	logarytm naturalny: ln(x), $x > 0$
log10(x)	logarytm o podstawie 10: $log_{10}(x)$, $x > 0$
pow(x,y)	x^y (błąd zakresu, gdy $x = 0$ i $y \leq 0$ lub, gdy $x < 0$ i y nie jest całkowite
sqrt(x)	$\sqrt{x}$, $x \geq 0$
ceil(x)	najmniejsza liczba całkowita nie mniejsza niż x
floor(x)	największa liczba całkowita nie większa niż x
fabs(x)	wartość bezwzględna $\|x\|$
ldexp(x,n)	$x \cdot 2^n$
frexp(x, int *exp)	rozdziela x na znormalizowaną część ułamkową z przedziału $[1/2, 1]$ i wykładnik potęgi 2; funkcja zwraca część ułamkową, a wykładnik potęgi wstawia się do *exp
modf(x, double *ip)	rozdziela x na część całkowitą i ułamkową, obie z tym samym znakiem, co x; część całkowitą wstawia się do *ip i zwraca część ułamkową
mod(x,y)	zmiennopozycyjna reszta z dzielenia x/y, z tym samym znakiem co x