Podstawy programowania.

Wykład 2

Zmienne i obsługa wejścia/wyjścia

Pierwszy program w C

```
/* korzystanie z biblioteki – komentarze... */
#include <stdio.h> // USES

// funkcja - "odpowiednik" algorytmu
// funkcja "main" - punkt wejścia, obecny w każdym programie
// konkretna funkcja main:
// brak danych wejściowych, w treści brak danych wyjściowych,
// wszystko jest efektem ubocznym
int main(void )
// nawiasy klamrowe wyróżniają grupy instrukcji - tutaj treść funkcji "main"
{
   // wcięcie, konwencje zapisu
   // printf - interakcja z systemem operacyjnym
   // standardowe wejście, standardowe wyjście, napis, znak nowej linii
   printf("hello, world\n"); // średnik
}
```

Kompilacja

- Kod źródłowy w postaci pliku tekstowego
- Pierwszym krokiem kompilacji jest wstępne przetworzenie pliku źródłowego (*preprocessing*)
- Preprocessing może obejmować:
  - automatyczne wstawienie do pliku zawartości innego pliku
    - "dyrektywa" preprocessingu: #include"*nazwa_pliku"*
    - inne pliki mogą dostarczać potrzebnych informacji np. o wykorzystywanych operacjach (funkcjach) systemu operacyjnego
      - wtedy można użyć #include<*nazwa_pliku*>
  - automatyczną zamianę zdefiniowanych symboli na odpowiadające wyrażenia
    - dyrektywa #define
    - np. #define SIEDEM 7
    - lepiej: #define LICZBA_DNI_TYGODNIA 7

Wejście/wyjście

- Obsługa konkretnego urządzenia wejścia/wyjścia jest realizowana przez system operacyjny
- Na poziomie języka (C i innych) pojawiają się abstrakcje umożliwiające korzystanie z wejścia/wyjścia bez znajomości szczegółów technicznych urządzeń
- Powszechną i użyteczną abstrakcją jest traktowanie interakcji z urządzeniem wejścia/wyjścia jako przesyłania strumieni znaków
  - podstawowy zestaw znaków (ASCII) nie obejmuje polskich liter
  - rozszerzone zestawy znaków (np. UTF) są obsługiwane przez nowsze kompilatory
  - pojedynczy znak UTF może zajmować więcej miejsca niż pojedynczy znak ASCII (jeden bajt)

Wejście/wyjście

- Język C posługuje się pojęciami standardowego wejścia i standardowego wyjścia
  - standardowym urządzeniem wejścia jest zazwyczaj klawiatura
  - standardowym urządzeniem wyjścia jest zazwyczaj terminal
    - w ramach graficznych interfejsów wykonywany program jest powiązany z konkretnym terminalem
    - brak terminala może zaburzać działanie programu
  - podstawowe funkcje obsługi strumieni wejścia i wyjścia pozwalają odczytać lub zapisać pojedynczy znak:
    - getchar() - uruchamia procedurę systemu operacyjnego pobierającą pojedynczy znak ze standardowego wejścia
    - putchar('a') - uruchamia procedurę systemu operacyjnego wypisującą pojedynczy znak na ekranie (w okienku) terminala

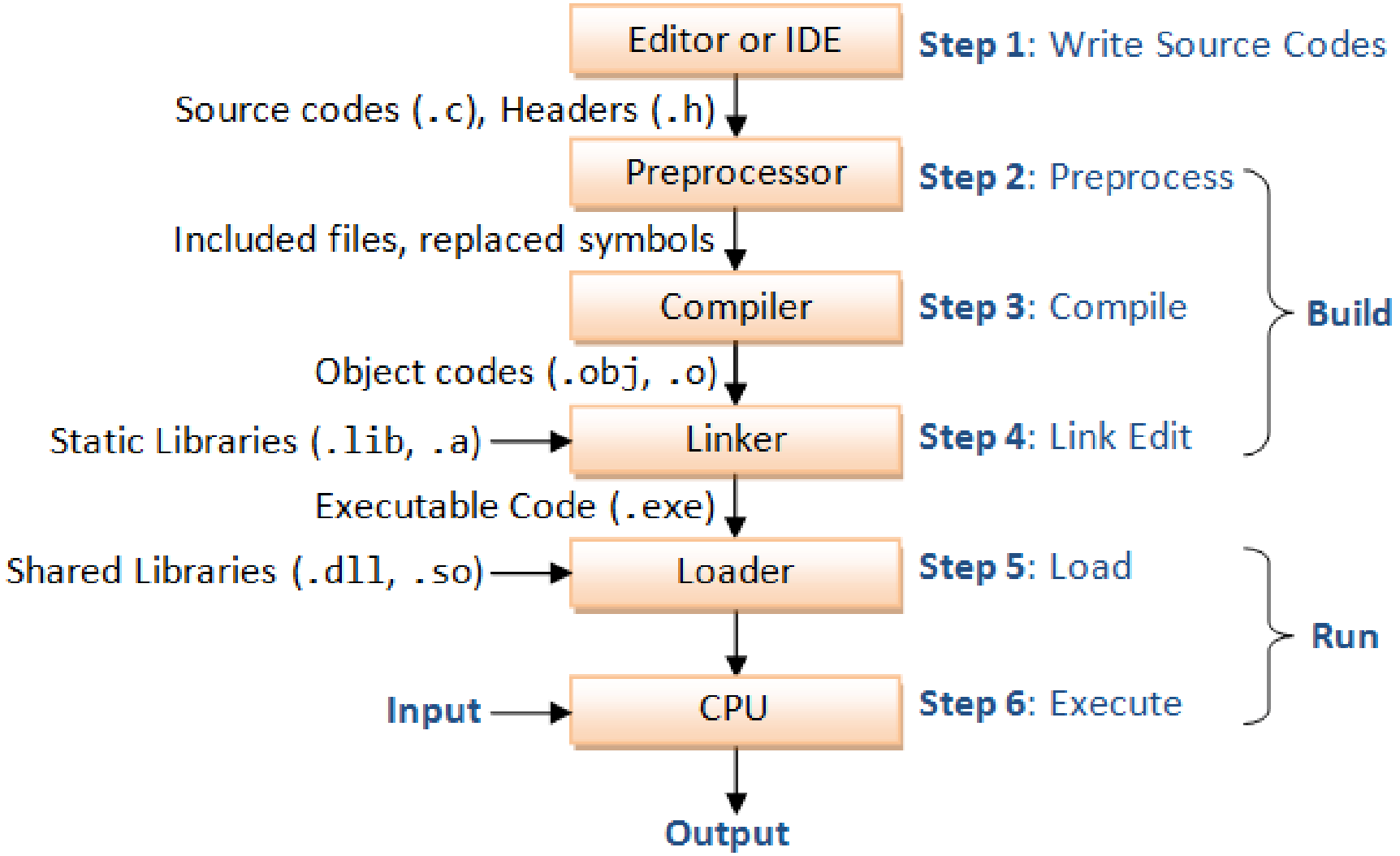
Wejście/wyjście

- Oprócz pobierania i wypisywania pojedynczych znaków, standard C udostępnia szereg operacji na napisach (ciągach znaków) i liczbach
  - printf(.....) - wypisuje ciąg znaków zgodnie z wzorcem formatowania przesyłanym jako jeden z argumentów (jedna z danych wejściowych)
    - nazwa printf pochodzi od określenia (*print formatted output*)
  - printf(".....") - wypisuje napis zawarty pomiędzy znakami cudzysłowu na ekranie terminala
  - scanf(....) - pobiera (odczytuje) ze standardowego wejścia dane zgodnie z wzorcem formatowania przesyłanym jako jeden z argumentów (jedna z danych wejściowych)
  - i wiele innych

Wejście/wyjście

- Informacje o wykorzystywanych funkcjach bibliotecznych zawarte są w tzw. plikach nagłówkowych
  - plikiem nagłówkowym zawierającym informacje o funkcjach wejścia/wyjścia jest plik stdio.h
- Informacje o realizowanych funkcjach zawarte w plikach nagłówkowych dołączane są do kodu źródłowego (formalnie wklejane) za pomocą dyrektywy #include
- Dzięki informacjom z plików nagłówkowych kompilator wie czy funkcje zostały poprawnie wykorzystane w programie
- Skompilowany kod źródłowy nie zawiera jeszcze skompilowanego kodu funkcji bibliotecznych
- Skompilowany kod funkcji bibliotecznych dołączany jest do ostatecznego programu na etapie linkowania

Tworzenie i wykonanie kodu



Programy = algorytmy + struktury danych

- Przebieg typowego programu:
  - wczytanie danych wejściowych
    - program rezerwuje miejsce na dane
    - wczytanie odbywa się przez wywołanie funkcji, których działanie zależy od sprzętu i systemu operacyjnego
  - realizacja przetwarzania
    - realizacja algorytmu wyrażonego w kodzie źródłowym instrukcjami operującymi na danych
    - w kodzie binarnym i języku asemblera instrukcje zamienione są na zestawy rozkazów procesora
  - zwrócenie wyniku, danych wyjściowych
    - poprzez wywołanie funkcji, których działanie zależy od sprzętu i systemu operacyjnego
- Istotna rola przechowywania danych (w pamięci operacyjnej) w trakcie wykonania programu

Programowanie

- Jak zapisać dane w pamięci operacyjnej?
  - typy danych (1, -5, 123456789, 2.6, 1/3, $2{,}34*10^{12}$, $1.9*10^{-15}$)?
- Typ danych – ujęcie abstrakcyjne
  - zestaw możliwych wartości, reprezentacja
  - zestaw możliwych operacji do wykonania na obiektach danego typu
- Typy danych – ujęcie uproszczone
  - wbudowane typy danych języka programowania, przykład C:
    - znaki (char) – jednobajtowe liczby całkowite
    - liczby całkowite (int) – rozmiar zależny od systemu (16, 32, 64 bity)
    - liczby rzeczywiste (zmiennoprzecinkowe)
      - pojedynczej precyzji (float) – 32 bity, 4 bajty
      - podwójnej precyzji (double) – 64 bity, 8 bajtów
    - wartości logiczne: true, false (od C99, wymaga stdbool.h)
      - zapisywane jako wartości całkowite (false==0, true==1)

Tablica kodów ASCII (7 bitów)

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	|
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

Zapis binarny

- Pozycyjne systemy liczbowe: dziesiętny, dwójkowy, szesnastkowy
- Przekształcanie do postaci binarnej i z postaci binarnej
- Zapis liczb naturalnych
  - kolejność bajtów (big-endian, little-endian)
- Zapis liczb całkowitych – ujemnych i dodatnich
  - ze znakiem (problem podwójnego zera) – zakres $(-2^{n-1}+1, 2^{n-1}-1)$
  - z zapisem od podstawy -2^n (zakres $(-2^{n-1}, 2^{n-1}-1)$ – problem 0 (zapis stosowany dla wykładnika liczb zmiennoprzecinkowych)
  - w systemie uzupełnień - zakres $(-2^{n-1}, 2^{n-1}-1)$, 0 jako wszystkie bity zerowe
- Dla liczb całkowitych reprezentacja binarna jest dokładna!
  - równość matematyczna jest identyczna z równością reprezentacji binarnej
  - jedynym problemem jest nieprzekraczanie dopuszczalnego zakresu wartości

Zapis binarny

- *Zapis liczb rzeczywistych*
  - zapis ułamków – rozszerzenie systemu pozycyjnego dwójkowego
    - dla liczb rzeczywistych tylko niektóre z nich dają się zapisać w postaci binarnej – pozostałe muszą być zaokrąglane
    - np. liczba 1.38 i jej zapis dwójkowy z czterema bitami po przecinku: 1.0110 (zapis z obcięciem 1.38 = 1.0110 + 0.005)
    - problem skończonej dokładności (np. liczba 1.0110 i brak liczb pomiędzy 1.3125, 1.375, 1.4375)
    - pojecie cyfr znaczących – przykład powyżej: jedna lub dwie
  - zapis wielkich i małych liczb - notacja wykładnicza („naukowa”)
    - liczba = znak * mantysa * $podstawa^{wykładnik}$
      - $-2{,}34*10^{12}$, $1.9*10^{-15}$
    - odpowiedniość przesuwania przecinka i zmiany wykładnika
    - dokładność zależna od liczby bitów mantysy (ok. 10 bitów na 3 cyfry znaczące)

Zapis binarny

- *Zapis liczb rzeczywistych*
  - formaty zmiennoprzecinkowe IEEE 754
  - pojedyncza precyzja, SP
    - 32 bity (znak + 23 bity mantysy + 8 bitów wykładnika)
  - podwójna precyzja, DP
    - 64 bity (znak + 52 bity mantysy + 11 bitów wykładnika)
  - mantysa znormalizowana (brak zapisu bitu przed przecinkiem)
    - zakres dla liczb dodatnich:
      - SP: od 1.2×10^{-38} do 3.4×10^{38}, DP: od 2.2×10^{-308} do 1.8×10^{308}
    - dokładność w cyfrach znaczących: SP – 6 do 9, DP – 15 do 17
  - operacje na liczbach zmiennoprzecinkowych IEEE 754
    - dodatkowa dokładność w trakcie realizacji operacji
    - pułapki i wyjątki (nadmiary, niedomiary, błędy operacji)
    - wartości specjalne (+/-0, +/-nieskończoność, NaN)

Zapis binarny

- *Zapis liczb rzeczywistych – specyfika i problemy*
  - (wydruki z 20 cyframi znaczącymi)
  - float f = 1.0/3.0; // (f = 0.33333343267440796)
    - tylko 7 pierwszych cyfr jest znaczących
  - double d = 1.0/3.0; // (d = 0.3333333333333315)
    - 16 cyfr znaczących
  - operacje na liczbach zmiennoprzecinkowych mogą dodatkowo zmniejszać liczbę cyfr znaczących (z powodu konieczności przekształcenia do tego samego wyznacznika przed operacją)
    - matematycznie: 1/0.0000123 = 81300,(81300)
    - double d = 1.0/0.0000123; (d = 81300.8130081300769)
      - 15 cyfr znaczących
    - double d = 1.0/((123000+0.0000123)-123000);
      - matematycznie identyczne z 1/0.0000123
      - w rzeczywistości (d = 81300.771057083708)
      - zostało tylko 5 cyfr znaczących!