Типове променливи. Преобразуване на типове променливи. ASCII. Побитови операции

I. Архитектура на Джон фон Нойман

По-голямата част от съвременните компютри работят въз основата на т.нар. архитектура на Джон фон Нойман.

Архитектурата на фон Нойман се състои от 4 основни компонента – входни устройства, памет, централен процесор и изходни устройства. Взаимодействието между тях обяснява как компютърът обработва данни.

1. Входните устройства на компютъра въвеждат в паметта данните, която трябва да бъдат обработени;
2. **При въвеждането на информацията в паметта се случва преобразуването ѝ в двоичен формат (binary format)***;
3. Централният процесор получава от паметта конвертираните в бинарен формат данни, обработва ги и отново записва в паметта вече обработените данни;
4. Изходните устройства изобразяват готовите данни от паметта в подходящ за потребителя вид.

По определение, думата „информатика" означава наука за събиране, съхранение, обработка и разпространение на данни. Може да запомните, че архитектурата на фон Нойман имплементира тези дейности с четирите си компонента - събиране на данни чрез входните устройства, съхранение (макар и кратковременно) на данни чрез паметта, обработка на данни чрез централния процесор и разпространение на данни чрез изходните устройства.

Обяснение на фон Нойманова архитектура:

https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:computers/xcae6f4a7ff015e7d:computer-components/v/khan-academy-and-codeorg-cpu-memory-input-output

https://www.youtube.com/watch?v=-SADbPS8UgA&ab_channel=MrKhan%27sClasses

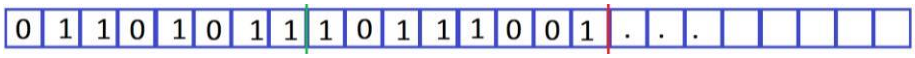
Памет. Паметта на компютъра може да се представи като редица от елементи 0 и 1, всеки от които е носител на информация (вж. *) .
*Тези елементи наричаме **bit-ове**.*

Технически не е възможно да се осъществи достъп до всеки такъв елемент на паметта, затова групираме битовете в по-големи информационни единици, наречени **машинна дума**. Големината на машинната дума варира (8, 16, 32 бита).

*8-битовата машинна дума се нарича **byte**.* Обемът на паметта се измерва в KB (2^{10}B), MB (2^{20}B), GB (2^{30}B), TB (2^{40}B), PB (2^{50}B) и т.н.

*Можем да достъпваме всяка дума в паметта. Тя се свързва с пореден номер, който наричаме неин **адрес**.***

Можем да си представим паметта като последователност от байтове, тоест последователност от 8-битови подпоследователности:



Информацията, която се записва в една такава подпоследователност (байт), се нарича ***стойност на клетката***. Всеки път, когато записваме нова стойност в такава клетка, старата се унищожава и не може да бъде възстановена. В тези клетки могат да се записват както данни, така и команди.

Полезно за паметта (на компютъра, не вашата ☺):
https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:computers/xcae6f4a7ff015e7d:computer-components/a/computer-memory

II. Променливи

В програмните езици данните се съхраняват в т.нар. **променливи (variables)**. Всяка променлива има:

1. Име: a, b, my_age, radius, etc. Това е уникален в рамките на програмата идентификатор, който си избираме за нашата променлива. Добрата практика изисква имената на променливите да са значещи (например my_age вместо a) и с консистентен стил на изписване (например всички са със snake case: my_age, или camel case: myAge/MyAge). По Google Style Guide приетият за C++ стил за именуване на променливи е snake case.

2. Стойност (value): a = 4.3, my_age = 69, etc. Стойността на променливата се записва в двоичен вид в паметта, която е била определена за тази променлива при задаването на тип и стойност на променливата.

3. Тип (type): цяло/десетично число, символ, булев, символен низ. Типът на променливата определяме и задаваме според стойностите, които искаме тя да приема, т.е. дефиниционната ѝ област (пр. ако искаме да приема само целочислени стойности, задаваме типа на променливата да бъде целочислен, int). Типовете на променливите са примитивни (число, символ, true/false) и съставни (изградени от много на брой примитивни).

4. Размер (size): броят байтове, които заема променливата в паметта;

5. Адрес (address): мястото в паметта, където се пази (поредния номер на машинната дума, с която променливата се свързва, вж. **).

III. Примитивни типове променливи в C++

Тип (type):	Размер (bit width):	Обхват (range):	Използва се за:
char	1byte	-127 to 127 or 0 to 255	Символи
unsigned char	1byte	0 to 255	Символи
short int (short)	2bytes	-32,768 to 32,767	Цели числа
unsigned short int	2bytes	0 to 65,535	Цели числа
int	4bytes	-2,147,483,648 to 2,147,483,647	Цели числа
unsigned int	4bytes	0 to 4,294,967,295	Цели числа
float	4bytes	+/- 3.4e +/- 38 (~7 digits)	Дробни числа
double	8bytes	+/- 1.7e +/- 308 (~15 digits)	Дробни числа
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)	Дробни числа
bool	1byte	true or false	Булеви числа (истина/лъжа)

Защо променливите имат такъв обхват?

https://www.youtube.com/watch?v=Bgs9PxHuF1M&list=PLNsrsUtqToOnydqvp2DT3R4noKC9BKRYB&index=31&t=195s&ab_channel=Velcode

IV. Създаване, въвеждане, извеждане на променливи

Създаване на променлива:

<тип на променливата> <име_на_променливата> {= <стойност>}$_{опц.}$

Примери: 1. да се създаде целочислена променлива my_age, която да пази възрастта Ви: int my_age = 19;

2. да се създаде десетична променлива P, която да няма първоначална стойност: double P; (може и float P).

Разлика между инициализация, декларация и дефиниция:

Декларация е, когато създаваме променлива, без да ѝ присвоим стойност, както в 2).

Инициализация е, когато присвояваме на променливата стойност за пръв път, както в 1). Дефиниция е, когато сме декларирали някъде променливата, но не сме ѝ задали стойност и след 100 реда код решаваме да ѝ зададем такава. Това не е добра практика! Добрата практика изисква да си създавате променливи непосредствено преди използването им.

Извеждане на конзолата: с оператора cout. (Иначе казано, с използване на изходния поток, за което ще учим малко по-нататък)

cout << <каквото_искаме_да_изведем>;

Пр.: 1. изведете на конзолата стойността на променливата my_age: cout << my_age;

2. изведете на конзолата на различни редове числата 13 и 31: cout << 13 << endl << 31;

Как се извежда нов ред: 2 начина – извеждаме **endl** или извеждаме **"\n".**

Пр.: а) cout << 13 << endl << 31; // извежда: 13
31

б) cout << 13 << 31;
cout << 45; // извежда: 133145

в) cout << 13 << "\n";
cout << 45; // извежда: 13
45

Въвеждане от потребителя: с оператора cin (входен поток)
– **cin >> <име_на_променлива> ;**

Пр.: да се състави програма, за която потребителят да въвежда цяло число и програмата да извежда числото, удвоено.

```
int number;
cin >> number;
cout << number * 2;
```

Присвояване на стойност на променлива: с оператора „="

```
Пр: int number = 4;
    cout << number; // 4
    number = 5;
    cout << number; //5
```

Равенство: ==

```
Пр: bool is_even = false;
is_even = 3 % 2;
cout << is_even == 1; //1
```

Размер на променлива: sizeof(<име_на_променливата>);

Пр. cout << sizeof(my_age); //извежда 4, защото сме my_age е от тип int и тип int има размер 4B;

Адрес на променлива: &<име_на_променливата>;

Пр. cout << &my_age; //извежда число в шестнадесетичен запис;

Коментар в C++: //<закоментиран_код> или **/*<закоментиран_код>*/ -** така отбелязваме, че даден откъс не трябва да се чете от компилатора и не участва в нашата програма;

V. Преобразуване на променливи

Неявно преобразуване:

```
double x = 2.3;
int y = x;
cout << y << " " << sizeof(y); //извежда: 2 4
```

Явно преобразуване: (тип)(<израз>)

```
int x = (int)(1.52 + 56.2);
double y = (double)(123 + 18);
cout << x << " " << y << " sizeof y: " << sizeof(y); //57 141 sizeof y: 8
```

Още едно явно преобразуване: static_cast<тип>(<променлива> или <израз>)

```
int x = static_cast<int>(1.52 + 56.2);
double y = static_cast<double>(123 + 18);
cout << x << " " << y << " sizeof y: " << sizeof(y); //57 141 sizeof y: 8
```

Когато преобразуваме double в int, int-ът приема цялата част на double – вж. 1. (аналогично с float).

VI. Символен тип (char) и ASCII

```
(1) char symbol = 'A';
(2) cout << symbol << endl;       //A
(3) cout << (int) symbol << endl; //65
(4) symbol = 'B';
(5) cout << (int) symbol << endl; //66
```

Създаване на променлива от тип char: ред (1). Забележете, че се използват единични кавички, а двойните са запазени за символни низове.

Как се пазят в паметта символните променливи? – по същия начин, по който и числовите. На всеки символ от ASCII таблицата съответства число.

ASCII таблица:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

За символния тип: https://www.youtube.com/watch?v=ozhU26jnToQ&ab_channel=CalebCurry

VII. Вградени функции и операции над числа

Операция	Оператор	Пример
Събиране	+	…
Изваждане	-	…
Умножение	*	…
Целочислено деление	/, операндите са int	3/5 == 0
Остатък при деление	%	7%2 == 1
Деление	/, поне един от операндите е double	(double)7/2 == 3.5
Степенуване	pow(x,n)	pow(3,2)==9
⌊x⌋	floor(x)	floor(3.5) == 3
⌈x⌉	ceil(x)	ceil(3.5) == 4
√x	sqrt(x)	sqrt(9) == 3
$\log_a b$	log(b)/log(a)	log(4)/log(2) == 2
Тригонометр. ф/ции	sin(x), cos(x), etc., където x е в радиани	sin(0) == 0

VIII. Логически операции

Операция	Оператор	Пример
Конюнкция (и)	&&	true && false == false
Дизюнкция (или)	||	true || false == true

IX. Побитови операции

Двоичната бройна система (също и бинарна система) е позиционна бройна система с основа 2, при която числата се изобразяват само с помощта на две цифри: 0 и 1.

На x_{10} в десетична бройна система съответства y_2 в двоична бройна система, първите 8 десетични числа изглеждат така:

- $1_{10} = 1_2$
- $2_{10} = 10_2$
- $3_{10} = 11_2$
- $4_{10} = 100_2$
- $5_{10} = 101_2$
- $6_{10} = 110_2$
- $7_{10} = 111_2$
- $8_{10} = 1000_2$

Когато трябва да обръщаме десетично число в двоично се процедира в следния ред:

1. Делим първоначалното число на 2;
2. Ако то се дели без остатък записваме 0;
3. Ако числото има остатък записваме 1;
4. Връщаме се отначало, докато не достигнем 0.

Например числото 19_{10} се преобразува по следния начин:

19	/ 2	= 9	с остатък 1
9	/ 2	= 4	с остатък 1
4	/ 2	= 2	с остатък 0
2	/ 2	= 1	с остатък 0
1	/ 2	= 0	с остатък 1

Остатъците се записват отдясно наляво. Така получаваме $19_{10} = 10011_2$.

Пример: от двоична в десетична БС:

$100101_2 = [(1) \times 2^5] + [(0) \times 2^4] + [(0) \times 2^3] + [(1) \times 2^2] + [(0) \times 2^1] + [(1) \times 2^0]$

$100101_2 = [1 \times 32] + [0 \times 16] + [0 \times 8] + [1 \times 4] + [0 \times 2] + [1 \times 1]$

$100101_2 = 3710$

Таблица на побитовите операции:

& (bitwise AND)	Взима две числа в двоичен запис и прилага AND на всеки бит от тях. Резултатът от операцията е 1 само ако битовете и на двете числа са единици.
| (bitwise OR)	Взима две числа в двоичен запис и прилага OR на всеки бит от тях. Резултът от операцията е 1, ако поне едното число има бит със стойност 1.
^ (bitwise XOR)	Взима две числа в двоичен запис и прилага операцията XOR на всеки бит от тях. Резултатът от операцията е 1 само ако битовете на двете числа са различни.
~ (bitwise NOT)	Взима едно число в двоичен запис и заменя всеки бит от него с противоположния му.
<< (left shift)	Взима едно число в двоичен запис и премества битовете му с X позиции наляво.
>> (right shift)	Взима едно число в двоичен запис и премества битовете му с X позиции надясно.

Пример:

```
#include<iostream>
using namespace std;
int main()
{
	char a = 5, b = 9;  // a = 5(00000101), b = 9(00001001)
	cout << "a&b=" << (a&b) << endl;  // The result is 00000001
	cout << "a|b=" << (a|b) << endl;  // The result is 00001101
	cout << "a^b=" << (a^b) << endl;  // The result is 00001100
	cout << "~a=" << (~a) << endl;    // The result is 11111010
	cout << "b<<1=" << (b<<1) << endl;// The result is 00010010
	cout << "b>>1=" << (b>>1) << endl;// The result is 00000100
	return 0;
}
```

Много готина поредица за C++ @Youtube : Velcode, Learn Programming with C++