Сибирь I

Типы, условия, циклы

Задача

Написать программу, проверяющую защищенность пароля

Каким требованиям должна соответствовать программа?

Обязательно:

- Принимать данные на вход
- Проверять символы введенного пароля
- Выводить ответ в зависимости от входных данных

Рубрика

Документация за 15 минут (40)

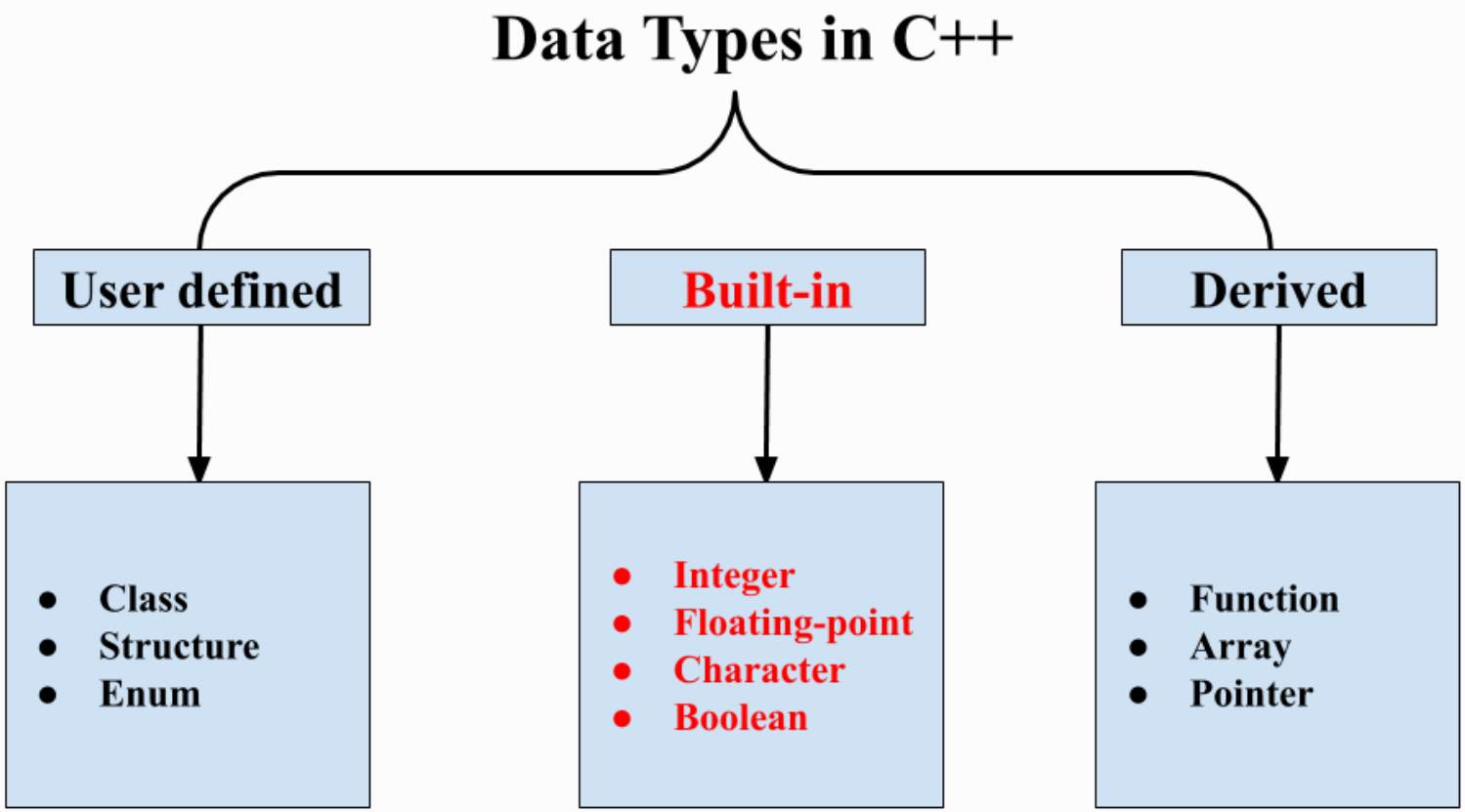
std::cin

```
std::cin >> <имя переменной>;    // ввод с клавиатуры

int x, y;
std::cin >> x >> y;

/* Это комментарий */
// Это - тоже
/// Видимый комментарий
```

Переменные

Переменная – это именованная область памяти

Типы данных


Data Types in C++
User defined
Built-in
Derived
Class
Structure
Enum
Integer
Floating-point
Character
Boolean
Function
Array
Pointer

Встроенные типы данных

Key word	Size in bytes	Interpretation	Possible values
bool	1	boolean	true and false
unsigned char	1	Unsigned character	0 to 255
char (or signed char)	1	Signed character	-128 to 127
wchar_t	2	Wide character (in windows, same as unsigned short)	0 to 2^{16}-1
short (or signed short)	2	Signed integer	-2^{15} to 2^{15} - 1
unsigned short	2	Unsigned short integer	0 to 2^{16}-1
int (or signed int)	4	Signed integer	-2^{31} to 2^{31} -1
unsigned int	4	Unsigned integer	0 to 2^{32} - 1
Long (or long int or signed long)	4	signed long integer	-2^{31} to 2^{31} -1
unsigned long	4	unsigned long integer	0 to 2^{32} -1
float	4	Signed single precision floating point (23 bits of significand, 8 bits of exponent, and 1 sign bit.)	$3.4*10^{-38}$ to $3.4*10^{38}$ (both positive and negative)
long long	8	Signed long long integer	-2^{63} to 2^{63} -1
unsigned long long	8	Unsigned long long integer	0 to 2^{64} -1
double	8	Signed double precision floating point(52 bits of significand, 11 bits of exponent, and 1 sign bit.)	$1.7*10^{-308}$ to $1.7*10^{308}$ (both positive and negative)
long double	8	Signed double precision floating point(52 bits of significand, 11 bits of exponent, and 1 sign bit.)	$1.7*10^{-308}$ to $1.7*10^{308}$ (both positive and negative)

ASCII (char == int?)

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Любую переменную встроенного типа данных *нужно инициализировать*, чтобы в ней не хранился мусор

Предпочтительнее использовать:

- int
- size_t (unsigned long)
- double
- char
- bool

Несколько примеров

```
int a = 0;
a += 3;     // эквивалентно a = a + 3;

// так же действуют операции -= *= /= %=

a /= 2;   // (a == 1) целочисленное деление

int b = sizeof(a); // (b == 4), sizeof возвращает размер типа

a %= b   // остаток от деления
```

Массивы

Массив — это область памяти, где могут последовательно храниться несколько значений *одного типа*.

Примеры

```
int numbers[4];
int numbers[4] = {1,2,3,4};
int numbers[4] = {1,2,3,4,5,6,7};          // ошибка
int numbers[] = {1,2,3,4,5,6};

char arr[] = {'u', 'w', 'u'}
char hello[] = "world";
```

Этого делать мы, конечно же, не будем

std::vector

Массив, опередивший свой размер

Примеры

```
#include <iostream>
#include <vector>
...
std::vector<int> vec0 = {1, 2, 3};
vec0.push_back(4);      //  добавляем элемент в конец вектора
vec0.resize(8);    // Изменяем размер вектора

std::vector<char> vec({'u', 'w', 'u'});
size_t vec_size = vec.size();
std::cout << vec[0] << vec[1] << vec[2];
```

std::string

Строка сына маминой подруги

Примеры

```
#include <iostream>
#include <string>
...
std::string str_a("Hello");
std::string str_b = "world";
std::string big_str = str_a + str_a[4] + ',';
big_str += str_b;

std::cout << big_str << " size = " << big_str.size();
```

Условия

```
...

if (condition) {
  ... // код при condition == true
} else {
  ... // код при condition == false
}

... //дальнейшее выполнение программы
```

Логические выражения

```
cond_a && cond_b // конъюнкция
cond_a || cond_b // дизъюнкция
cond_a == cond_b // эквивалентность
cond_a != cond_b // не равно
!cond_a // отрицание
>   >=    <    <= // математика
```

Циклы

Цикл – это повторение одного и того же действия раз за разом в надежде на конечность цикла

while () {}

```
while (condition) {
  ...
  if (flag) {
    break; // позволяет выйти из цикла досрочно
  }
  ...
}
... // Начинается, как только condition == true
```

for () {}

```
...

for (size_t i = 0; i < n; ++i) {
  ... // тело цикла
}

...

for (;;) {
  ... // выхода нет
}
```

LET’S PARTY