Глава 6 «Строки. Массивы символов»

Библиотека string

Библиотека string в C++ предоставляет возможность работать со **строками**. Она позволяет создавать, изменять, обрабатывать и управлять ими, предоставляя удобные методы для работы с текстом. Чтобы можно было использовать эту библиотеку, необходимо **подключить заголовочный файл <string>** в начале программы **перед функцией main().**

```
#include <string>
```

Подключение библиотеки string к программе.

Тип данных string

String в C++ – это тип данных, представляющее собой **последовательность символов (строку),** используемую для хранения и обработки текста. Он определен в стандартной библиотеке C++ **<string>.**
Чтобы объявить переменную, нужно как обычно указать её тип (в данном случае string) и имя. Инициализировать можно как при объявлении, так и после.

```
string str = "";
```

Объявление переменной str со значением пустой строки.

```
string str2 ="Hello";
```

Объявление переменной str2 со значением строки Hello.

```
string str3;
str3 = "I love C++!";
```

Объявление переменной str3 со значением строки I love C++. Инициализация происходит позже.

Тип данных char

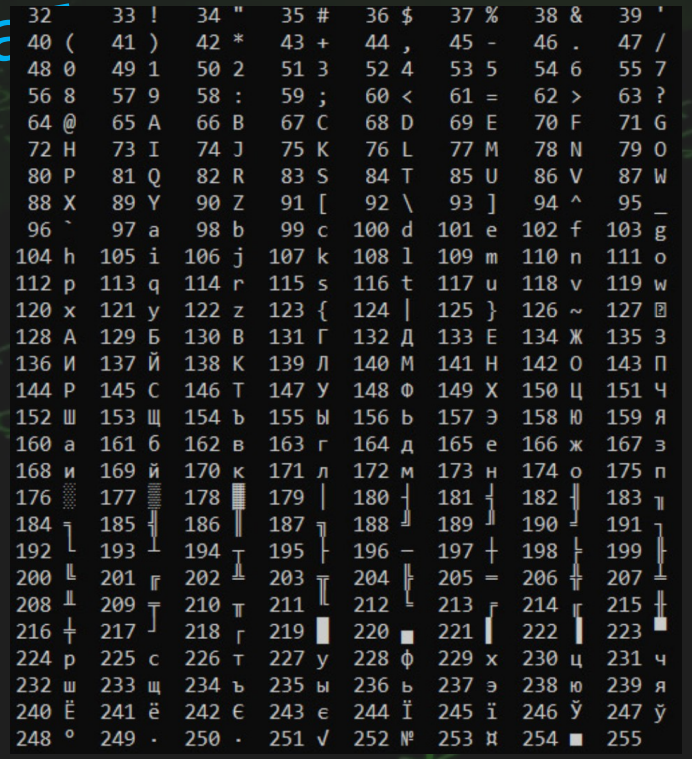
Тип char используется для хранения **одиночных символов**, таких как буквы, цифры и знаки препинания. Он занимает 1 байт (8 бит) в памяти и может представлять символы из различных кодировок, чаще всего используется **ASCII.** Объявление аналогично, за исключением того, что при инициализации символы указываются в **одинарных кавычках (' ').**

```
char ch = 'a';
```

Объявление переменной ch
со значением символа a.

char. Символы как числа

Также **тип char** может использоваться для хранения **целых чисел** (от 0 до 255), так как каждому символу соответствует его **ASCII-код.** Например символу‘0’соответствует код 48, а ~ код 126. Полная та



32		33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	⌂
128	А	129	Б	130	В	131	Г	132	Д	133	Е	134	Ж	135	З
136	И	137	Й	138	К	139	Л	140	М	141	Н	142	О	143	П
144	Р	145	С	146	Т	147	У	148	Ф	149	Х	150	Ц	151	Ч
152	Ш	153	Щ	154	Ъ	155	Ы	156	Ь	157	Э	158	Ю	159	Я
160	а	161	б	162	в	163	г	164	д	165	е	166	ж	167	з
168	и	169	й	170	к	171	л	172	м	173	н	174	о	175	п
176	░	177	▒	178	▓	179	│	180	┤	181	╡	182	╢	183	╖
184	╕	185	╣	186	║	187	╗	188	╝	189	╜	190	╛	191	┐
192	└	193	┴	194	┬	195	├	196	─	197	┼	198	╞	199	╟
200	╚	201	╔	202	╩	203	╦	204	╠	205	═	206	╬	207	╧
208	╨	209	╤	210	╥	211	╙	212	╘	213	╒	214	╓	215	╫
216	╪	217	┘	218	┌	219	█	220	▄	221	▌	222	▐	223	▀
224	р	225	с	226	т	227	у	228	ф	229	х	230	ц	231	ч
232	ш	233	щ	234	ъ	235	ы	236	ь	237	э	238	ю	239	я
240	Ё	241	ё	242	Є	243	є	244	Ї	245	ї	246	Ў	247	ў
248	°	249	∙	250	·	251	√	252	№	253	¤	254	■	255	

```
char ch = '~';
int x = ch;
cout << x;
```

Вывод

126

Данный фрагмент кода создает переменную char со значением ~ и присваивает целочисленной переменной x. В этом случае переменной x присваивается ASCII код символа, в данном случае он равен 126.

Массивы символов

В C++ **символьный массив** представляет собой **непрерывный блок памяти**, где каждый элемент представляет один символ. Каждый символ занимает **один байт** памяти, что позволяет напрямую обращаться к отдельным символам и манипулировать ими. Отличительная особенность в том, что при инициализации мы **можем передать** символьному массиву как **н**

```
char hello1[] {'h', 'e', 'l', 'l', 'o'};
char hello2[] {"hello"};
```

ку.

Первый массив hello1 инициализируется с помощью отдельных символов char, а второй hello2 с помощью строки string

Обработка строк. Операции над строками.

Со строками и символами можно выполнять следующие операции:

= – присвоить значение переменной.

```
string s;
s = "Hello";
```

Создается переменная s и присваивается значение Hello

+= – добавить в конец строки другой строки или символа.

```
string str1 = "He";
str1 += "llo";
cout << str1;
```

Вывод
Hello

Создается строка str1 со значением He, к ней добавляется строка llo, в результате получаем строку Hello.

Обработка строк. Операции над строками.

+ – конкатенация (сложение) двух строк или строки и символа.

```
string str1 = "Hello";
string str2 = "Would!";
string str3 = str1 + " "+ str2;
cout << str3;
```

Вывод

```
Hello Would!
```

Создаются 2 строки str1 и str2 со значениями Hello и Would! соответственно. В результате сложения получаем новую строку str3 со значением Hello Would!.

```
string str1 = "Hello";
char ch = '!';
string str2 = str1 + ch;
cout << str2;
```

Вывод

```
Hello!
```

Создаются строка str1 и символ ch со значениями Hello и ! соответственно. В результате сложения получаем новую строку str2 со значением Hello!.

== Посимвольное сравнение строк (true – если символы совпадают).

```
string str1 = "Hello";
string str2 = "Hello";
if (str1 == str2){

}
```

Логическое выражение = true, так как строки полностью совпадают.

```
string str1 = "Hello";
string str2 = "C++";
if (str1 == str2){

}
```

Логическое выражение = false, так как строки не совпадают.

Обработка строк. Операции над строками.

!= Посимвольное сравнение строк (true если хотя бы один символ не совпадает)

```
string str1 = "Hello";
string str2 = "Hello";
if (str1 != str2){

}
```

Логическое выражение = false, так как строки полностью совпадают.

```
string str1 = "Hello";
string str2 = "C++";
if (str1 != str2){

}
```

Логическое выражение = true, так как строки не совпадают.

<, >, <=, >= – лексикографическое сравнение строк (по первому символу по коду ASCII, если он равен, то по второму символу и так далее до конца строки. Если длина строк разная, а символы все совпадают, то строка с меньшим количеством символов меньше.

Обработка строк. Операции над строками.

```
string str1 = "Hello";
string str2 = "C++";
if (str1 > str2) {
}
```

Логическое выражение = true, так как первый символ H (ASCII код 72) больше символа C (ASCII код 67).

```
string str1 = "Hello";
string str2 = "C++";
if (str1 < str2) {
}
```

Логическое выражение = false, так как первый символ H (ASCII код 72) больше символа C (ASCII код 67).

```
string str1 = "Hello";
string str2 = "Hellowould";
if (str1 >= str2) {
}
```

Логическое выражение = false, так как str2 целиком содержится в str1.

```
string str1 = "Hello";
string str2 = "Hellowould";
if (str1 <= str2) {
}
```

Логическое выражение = true, так как str2 целиком содержится в str1.

escape-последовательности

В C++ **escape-последовательности** используются для **представления символов, которые не могут быть напрямую введены в строку** или имеют специальное значение. Они начинаются с **обратного слэша (`\`)** и заменяются соответствующими символами при компиляции.

Вот основные escape-последовательности:
\n: Перевод строки.
\t: Горизонтальная табуляция.
\v: Вертикальная табуляция.
\b: Забой (возврат на одну позицию).
\r: Возврат каретки.
\f: Прогон страницы.
\\: Обратная косая черта.
\': Апостроф.
\": Кавычка.
\?: Знак вопроса.
\0: Нулевой символ (конец строки).

Функции getline() и length()

Функция **getline()** используется для **чтения** строки текста из **входного потока** (например, клавиатуры или файла) и **сохранения** ее в **строковой переменной.** Она особенно полезна, когда вам нужно прочитать строки, содержащие пробелы.

Функция **length() возвращает длину строки** (количество символов в это

```
string str = "Hello";
int n = str.length();
cout << n;
```

Вывод
5

Строка Hello содержит 5 символов, её длина равна переменной n, то есть 5

Функции clear() и replace()

Функция **clear()** очищает строку (делая её пустой), а **replace()** позволяет заменить любой элемент в строке (по индексу элемента в строке).