Введение в язык программирования C++.

Элементы языка общие с C.

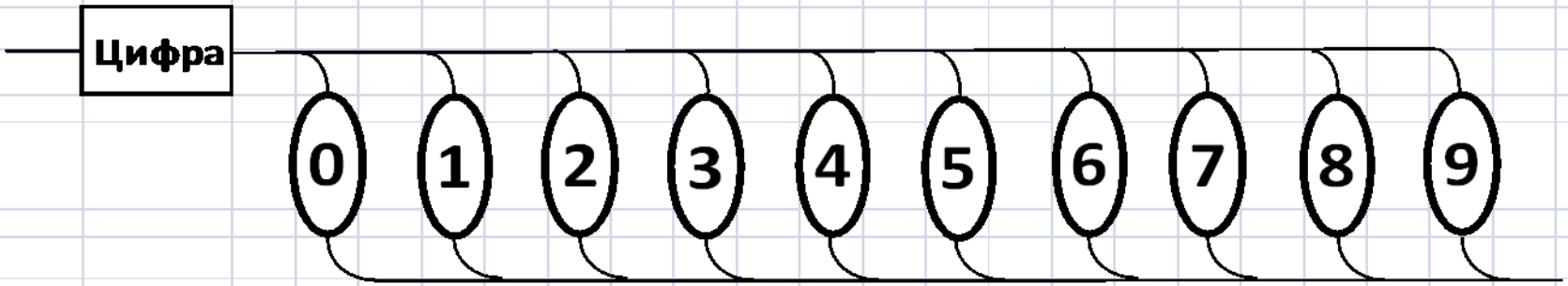
Полиморфизм.

Инкапсуляция.

История

- 1970 – Dennis Ritchie разработал язык C для DEC PDP-11 в ОС UNIX.
- 1979 – Bjarne Stroustrup в Bell Laboratory разработал язык C++.
- 1989 – American National Standart Institute утвердил стандарт, который поэтому называется ANSI.
- 1998 – International Standards Organization ратифицирован стандарт языка.

Алфавит языка

- <Буква> ::= A|B| … |Y|Z|a|b| … |y|z|_
- 

- Символы:

  +, -, *, /, >, =, <, !, &, |, ., \,,’, “, ?, :, ~, {, }, %, ^, (,), [,], ; , запятая

Терминальные символы
(Зарезервированные слова)

asm	default	for	operator	sizeof	unsigned
auto	delete	friend	private	static	virtual
break	do	goto	protected	struct	void
case	double	if	public	switch	volatile
char	else	inline	register	template	while
class	enum	int	return	this	
const	extern	long	short	typedef	
continue	float	new	signed	union	

Идентификатор –

::= <Буква> |
<Идентификатор><Буква> |
<Идентификатор><Цифра>

Допустимые:

count

Test23

Ptr_1

Недопустимые:

1count

sizeof

Ptr…1

Переменная - это

- Имя (идентификатор) – адрес в памяти машины;
- Тип – размер (количество байт) памяти машины;
- Значение – вид набора бит, задающих ее конкретную величину.

Базовые типы данных и литеральные константы

Целые длиной один байт – char

'0'

ASCII - American National Standard Code for Information Interchange

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht	nl	vt	np	cr	so	si
10	dle	dcl	dc2	dc3	dc4	nak	syn	etb	can	em	sub	esc	fs	gs	rs	us
20	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	del

Специальные символьные константы

\a	Сигнал	\t	Горизонтальная табуляция
\f	Подача бумаги	\v	Вертикальная табуляция
\n	Новая строка	\’	Одинарная кавычка
\r	Возврат каретки	\xN	Шестнадцатеричная или восьмеричная константа
\?	Знак вопроса	\N	
\”	Двойная кавычка	\b	Удаление предыдущего символа
\\	Обратный слеш		

Базовые типы данных и литеральные константы

Целые длиной один байт – char

'0'

Целые длиной несколько байт – int

48 060 0x30

С плавающей точкой – float

48.f .48e2F

Двойной точности - double

48. +480.0e-1

Без значения - void

Модификация базовых типов

Со знаком – signed

Без знака – unsigned

Короткое – short

Длинное – long

$$1 = \text{sizeof(char)} \le \text{sizeof(short)} \le$$
$$\le \text{sizeof(int)} \le \text{sizeof(long)}$$

$$\text{sizeof(float)} \le \text{sizeof(double)} \le$$
$$\le \text{sizeof(long double)}$$

Пример:

cin – стандартный ввод (>>)
cout – стандартный вывод (<<)

```
#include <iostream.h>
char c='\t';
short s=1;
int i;
long l;
void main()
{   float f;
    double d;
    long double D;
    cout << sizeof(c) << ' ';
    cout << sizeof(s) << ' ';
    cout << sizeof(i) << ' ';
    cout << sizeof(l) << ' ';
    cout << sizeof(f) << ' ';
    cout << sizeof(d) << ' ';
    cout << sizeof(D) << '\n';
}
```

Глобальные переменные

Локальные автоматические переменные

Результат выполнения программы: 1 2 2 4 4 8 10

Спецификаторы класса памяти

auto - Локальная переменная не существует за пределами области видимости.

static - Значение переменной постоянно хранится внутри функции или файла.

register - Обеспечить доступ к объекту так быстро, как только возможно.

extern - Объявляемая переменная определена в другой части программы.

```
extern int x;        static int y=0;
```

Квалификаторы типа

- const – неизменяемые данные

  Переменная не может программно изменять своего значения

const float e=2.71;

- volatile – подавление оптимизации

  Переменная может изменяться независимо от программы.

NB: порядок вычисления выражений не определен!

X = X1 + X2;

Оператор присваивания

Lvalue = Rvalue
Lvalue и Rvalue - идентификаторы

Lvalue – именующее выражение (адрес переменной)
Rvalue – вычисляемое выражение (значение переменной)

Множественное присваивание

```
x = y = z = 0;
```

Стенографическое присваивание

```
x ☼ = 10;    // x=x ☼ 10;
```

Арифметические операции

-	Вычитание	*Пример*
+	Сложение	float y,x = 5/3; //=1.0
*	Умножение	y = 5./3; // =1.666…
/	Деление	int k=1;
%	Остаток от деления	++k; // =2
--	Декремент	int m=--k; // k--, m=k;
++	Инкремент	int n=k++; // n=k, k++;
		// m=1, n=1

Операции сравнения и логические операции

>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
!=	Не равно

&&	И
||	ИЛИ
!	НЕ

Пример

```
int a,b,c,x=5,y=0;
a=x&&y; // = false
b=x||y;      // = true
c=!x;        // = false
```

Поразрядные операции

&	И	int x=6,y=3; //…0110,…0011
|	ИЛИ	x & y; // 0110&0011=2
^	Сложение по модулю два	x | y; // 0110|0011=7
		x ^ y; // 0110^0011=5
~	Дополнение к 1	~y; // = 1…100 = -4
>>	Сдвиг вправо	x>>1; // 0110>>1=0011=3
<<	Сдвиг влево	x<<1; // 0110<<1=1100=12

Еще несколько операторов

?: - условное выражение

```
y = x >= 0 ? 1 : -1;
```

sizeof - определение размера операнда в байтах

```
I = sizeof(int);
J = sizeof I;
```

, - последовательного вычисления

```
int k=(y=J, 2*y);
```

(*тип*) - явное преобразование к заданному типу

```
float z=(float)5/3;
z=float(5)/3;
```

:: - расширения области видимости

```
int i;
void main() { int i;
i=1; ::i=2; …
```

Массивы

Const int *размер* = ...;

Тип имя_переменной [*размер*];

int A[10];

Номер элемента	1-й	2-й	3-й	4-й	…
Имя элемента	A[0]	A[1]	A[2]	A[3]	…
Адрес элемента	0108	010a	010c	010e	…

int B[5][10]; B[3][4] – верно

A[1] – неверно

Указатели и ссылки

Указатель –
переменная, значением которой является адрес.

*P

Ссылка –
другое имя или еще одно имя объекта.

&P

NB: Может быть только инициализирована.

```
int x[2],*Ptr;
Ptr = &x[0];
*Ptr++=1; *Ptr=2;
```

```
int x[]={0,0},&b=x[0];
++b+=1;
```

```
cout << x[0] << ‘ ‘ << x[1] << ‘\n’;
```

Результат: 1 2 2 0

Операторы выделения и освобождения памяти

- void* new(size_t) – возвращает адрес непрерывного участка памяти для объекта типа и размерностью size_t байт.
- void delete(void*) – освобождает память, выделенную оператором new, начиная с адреса, на который ссылается указатель.

Пример.

```
const int ar_sz=10;
void main() {
float* ptr = new float[ar_sz];
ptr[1]=1.f; ptr[2]=2.f;
delete ptr;
}
```

Массивы и указатели

Описание

int x[]={1,2,3,4,5}; int x[5]; const int *y=x;

Адрес элемента массива

&x[3] y+3

Значение элемента массива

x[3] *(y+3)

Имя массива – неизменяемый указатель, инициализированный начальным значением.

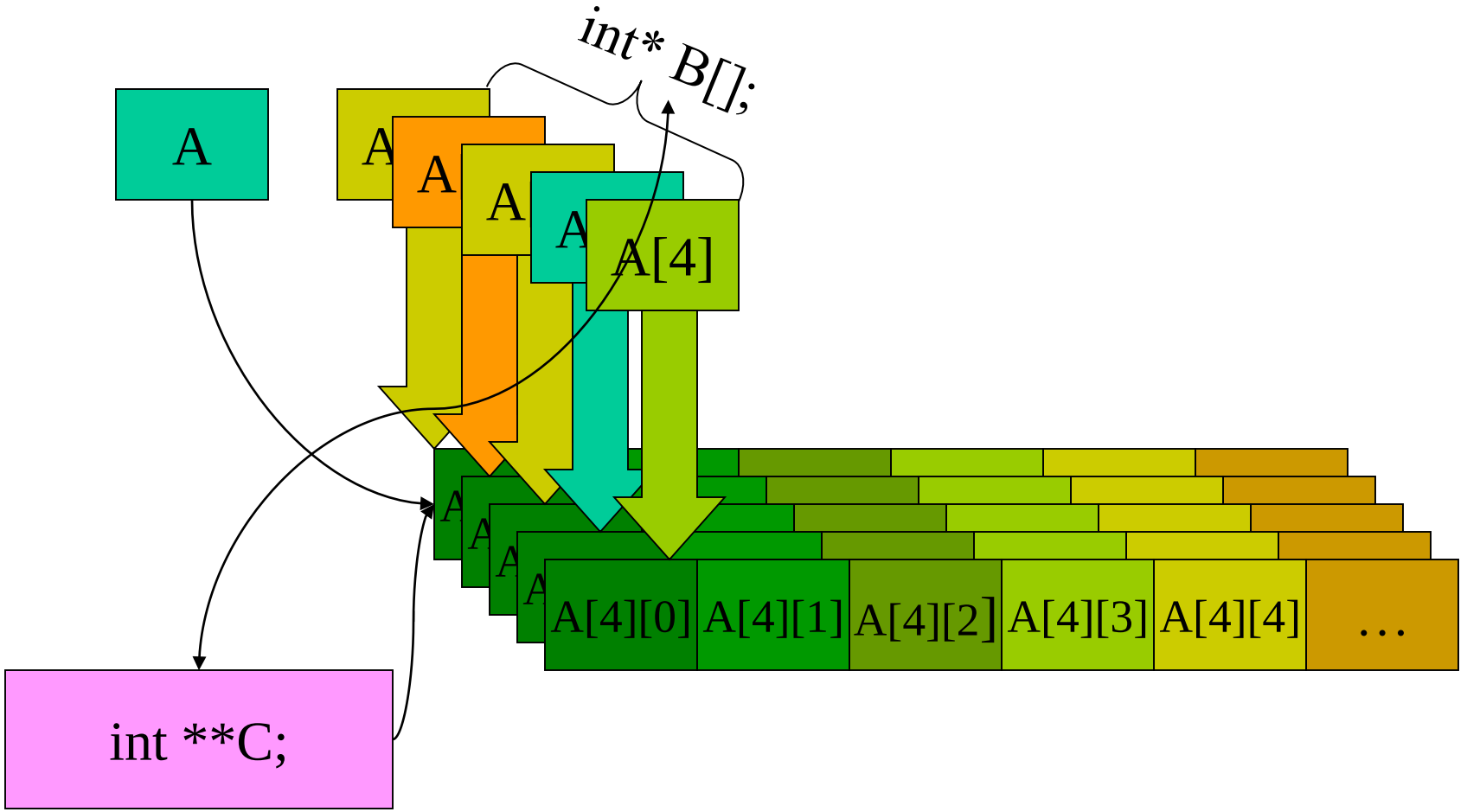
Строки

- одномерные массивы символов, заканчивающиеся символом с кодом ноль (‘\0’)

```
const int k=…;
char str[k];                    // это ещё не строка
str[0]=‘H’; str[1]=‘e’; str[2]=str[3]=‘l’;
str[4]=‘o’; str[5]=‘\0’;
char str0[]={‘H’,’e’,’l’,’l’,’o’,’\0’};
char* str1=“Hello”;
```

Двумерные массивы и двойные указатели

```
const int ik=5, jk=10;
void main()
{  int a[ik][jk];
   int* b[]={a[0],a[1],a[2],a[3],a[4]};
   int **c=b;  int i=4, j=4;
   *(*b + i*jk +j)=10;
   cout << *(*(c+i)+j) << '\n'; }
```


int* B[];
A
A
A
A
A
A[4]
A[4][0]
A[4][1]
A[4][2]
A[4][3]
A[4][4]
…
int **C;

Составные типы данных

- Структура – конгломерат переменных
- Объединение – наложение переменных
- Битовое поле – организация доступа к отдельным битам
- Перечисление – список именованных целых констант
- Имена типов, введенные пользователем:

  typedef *тип новое_имя*;

КОНГЛОМЕРАТ – механическое соединение чего-либо разнородного, беспорядочная смесь.

Словарь иностранных слов.

Структуры

- Формат описания:

struct *имя_шаблона* {
тип имя_члена;
…
тип имя_члена; }
список_переменных;

- Пример

struct complex {
float re,im;} a;

a

Имя_шаблона = тег

- Массивы структур:

struct complex d[5];

- Доступ к членам:

complex c,*b=&c;
a.re=0.; d[1].im=3.;
*b.re=1.; b–>im=2.;

- Присваивание:

c=*b;

Операции доступа к члену структуры:
. – через переменную
-> - через указатель

Объединения

- Формат описания:

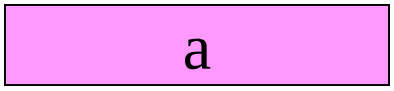
union *тег* {

тип *имя_члена*;

…

тип *имя_члена*; }

список_переменных;



- Пример:

```
union {
    int i;
    struct {char h_b,l_b;} j;
    } a;
a.j.h_b = '\0'; a.j.l_b='1';
cout << a.i << '\n';
```

Результат: 49

Битовые поля – члены структуры или объединения

- Формат:

struct или union *тег* {

тип *имя: длина*;

…

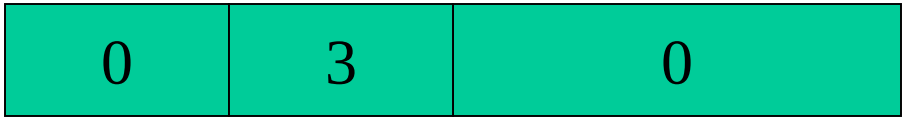
тип *имя: длина*; }

список_переменных;

- Здесь тип – это int, signed, unsigned

- Пример:

```
struct { int a:4; int b:4;
         char c; } c;
c.a=0; c.b=3; c.c='\0';
void* ptr=&c;
cout << *((int*)ptr);
```



Перечисление – набор именованных целых констант

- Формат:
  enum *тег* {
  список_перечисления
  }
  список_переменных;
- Пример

```
enum {m,t,w,th,f,s,su} d;
cout << m << ‘ ’ << su;
```

Результат: 0 6

- Пример

```
enum rmsk { I=1, V=5,
    X=10, L=50, C=100,
    D=500, M=1000 } i;
i=M+D;
cout << i << ‘\n’;
```

Результат: 1500

Приоритеты операций

Наивысший	::		== !=
	() [] -> .		&
	! ~ ++ -- - + (type) & sizeof * new delete		^
			|
	* / %		&&
	+ -		||
	<< >>		?:
	< <= >= >	Наинизший	= ☼=