Семинар 3. Знакомство с языком С.

1. Первая программа и команда

1.1. Создание, компиляция и выполнение первой программы.

Текст первой программы можно создать в терминальном окне с помощью редактора vi:

```
>vi hello.c
```

Просмотреть текст программы можно командой cat:

```
>cat hello.c
#include <stdio.h>

int main()
{
  printf("Hello!\n");
  printf("Привет!"\n");

  return 0;
}
```

Откомпилировать программу можно с помощью компилятора GNU C:

```
> gcc -o hello hello.c
```

Как проверить, что исполняемая программа появилась:

```
>ls -l
total 20
-rwxr-xr-x 1 mephi99 mephi 12264 Oct  7 11:09 hello
-rw-r--r-- 1 mephi99 mephi    70 Oct  7 11:09 hello.c
```

Выполнить готовую программу можно с помощью команды:

```
>hello
Hello!
Привет!
```

1.2. Моя первая команда

Попробуем создать на основе программы новую команду:

```
>vi mycmd.c
>cat mycmd.c
#include <stdio.h>
int main(int argc, char *argv[])
{
  int i;
  printf("mycmd: argc=%d\n", argc);
  for (i=0; i<argc; i++)
    printf("mycmd: argv[%d]=%s\n",i,argv[i]);
  return 0;
}
>gcc -o mycmd mycmd.c
>mycmd
mycmd: argc=1
mycmd: argv[0]=mycmd
>mycmd Hello
mycmd: argc=2
mycmd: argv[0]=mycmd
mycmd: argv[1]=Hello
>mycmd Hello vsem
mycmd: argc=3
mycmd: argv[0]=mycmd
mycmd: argv[1]=Hello
mycmd: argv[2]=vsem
>mycmd "Привет всем!"
mycmd: argc=2
mycmd: argv[0]=mycmd
mycmd: argv[1]=Привет всем!
```

1.3. Мой строчный калькулятор

```
Создание:
>vi mycalc.c
Просмотр:
>cat mycalc.c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  int i,a,b,c;
  printf("mycalc: argc=%d\n", argc);
  for (i=0; i<argc; i++)
    printf("mycalc: argv[%d]=%s\n",i,argv[i]);

  if (argc > 3) if (strlen(argv[2]) == 1) {
    sscanf(argv[1],"%d",&a);
    sscanf(argv[3],"%d",&b);
    if (argv[2][0] == '+') { c = a + b; printf("a+b=%d\n", c);}
    if (argv[2][0] == '-') { c = a - b; printf("a-b=%d\n", c);}
    if (argv[2][0] == '*') { c = a * b; printf("a*b=%d\n", c);}
    if (argv[2][0] == '/') { if (b == 0) c = 0; else c = a / b; printf("a/b=%d\n", c);}

  }

  return 0;
}

Результаты выполнения:
> mycalc 10 "+" 15
mycalc: argc=4
mycalc: argv[0]=mycalc
mycalc: argv[1]=10
mycalc: argv[2]=+
mycalc: argv[3]=15
a+b=25

> mycalc 10 "-" 15
mycalc: argc=4
mycalc: argv[0]=mycalc
mycmd: argv[1]=10l
mycmd: argv[2]=-
mycmd: argv[3]=15
a-b=-5

> mycalc 10 "*" 15
mycalc: argc=4
mycalc: argv[0]=mycalc
mycalc: argv[1]=10
mycalc: argv[2]=*
mycalc: argv[3]=15
a*b=150

> mycalc 10 "/" 15
mycalc: argc=4
mycalc: argv[0]=mycalc
mycalc: argv[1]=10
mycalc: argv[2]=/
mycalc: argv[3]=15
a/b=0
> mycalc 30 "/" 15
mycalc: argc=4
mycalc: argv[0]=mycalc
mycalc: argv[1]=30
mycalc: argv[2]=/
mycalc: argv[3]=15
a/b=2
```

2. Общая структура программы

2.1. Алфавит

Алфавит составляют буквы латинского алфавита (большие и маленькие), арабские цифры, пробелы и знаки табуляции, разделители.

Символ	Наименование	Символ	Наименование
,	Запятая	!	Восклицательный знак
.	Точка	|	Вертикальная черта
;	Точка с запятой	/	Наклонная черта вправо (слэш)
:	Двоеточие	\	Наклонная черта влево(обратный слэш)
?	Знак вопроса	~	Тильда
'	Одиночная кавычка (апостроф)	_	Подчеркивание
(	Левая круглая скобка	#	Знак номера
)	Правая круглая скобка	%	Процент
{	Левая фигурная скобка	&	Амперсанд
}	Правая фигурная скобка	^	Стрелка вверх
<	Знак "меньше"	-	Знак минус
>	Знак "больше"	=	Знак равенства
[	Левая квадратная скобка	+	Знак плюс
]	Правая квадратная скобка	*	Знак умножения (звездочка)

2.2. Специальные символы

Специальный символ	Шестнадцатеричное значение в коде ASCII	Наименование
\n	0A	Новая строка
\t	09	Горизонтальная табуляция
\v	0B	Вертикальная табуляция
\b	08	Забой
\r	0D	Возврат каретки
\f	0C	Новая страница
\a	07	Звуковой сигнал
\'	2C	Апостроф
\"	22	Двойная кавычка
\\	5C	Обратный слэш
\ddd		Байтовое значение в восьмеричном представлении
\xdd		Байтовое значение в шестнадцатеричном представлении

2.3. Операции

Операция	Наименование	Операция	Наименование
!	Логическое НЕ	^	Поразрядное исключающее ИЛИ
~	Обратный код	&&	Логическое И
+	Сложение; унарный плюс	||	Логическое ИЛИ
-	Вычитание; унарный минус	?:	Условная операция
*	Умножение; косвенная адресация	++	Инкремент
/	Деление	--	Декремент
%	Остаток от деления	=	Простое присваивание
<<	Сдвиг влево	+=	Присваивание со сложением
>>	Сдвиг вправо	-=	Присваивание с вычитанием
<	Меньше	*=	Присваивание с умножением
<=	Меньше или равно	/=	Присваивание с делением
>	Больше	%=	Присваивание с остатком от деления
>=	Больше или равно	>>=	Присваивание со сдвигом вправо
==	Равно	<<=	Присваивание со сдвигом влево
!=	Не равно	&=	Присваивание с поразрядным И
&	Поразрядное И; адресация	|=	Присваивание с поразрядным включающим ИЛИ
|	Поразрядное включающее ИЛИ	^=	Присваивание с поразрядным исключающим ИЛИ
,	Последовательное выполнение (запятая)		

2.4. Основные зарезервированные слова

```
bool break                                  int long
case char const continue                    return
default do double                           short signed sizeof static struct
else enum extern                            switch true typedef
false float for                             union unsigned void
goto if                                     while
```

2.5. Пример структуры программы

```
Специальные включения:
#include <stdio.h> // заголовки функций ввода-вывода
#include <stdlib.h> // заголовки функций стандартной обработки данных
#include <string.h> // заголовки функций обработки строк
#include <math.h> // заголовки математических функций

// Специальные определения:
#define Pi 3.1415926535897932384 // определение констант
#define myint long long int // переопределение базовых названий

// Макросы:
#define TRANSLATE_BOX_COORDS_01(bx0, by0, bz0, bx1, by1, bz1) \
{ \
  bx1 = (bx0) - 1; \
  by1 = (by0) - 1; \
  bz1 = (bz0) - 1; \
}

// Объявления новых типов:
typedef struct tag_int_Ar_1D_t {
  int len;
  int max;
  int *data;
} int_Ar_1D_t;

// Объявления глобальных переменных:
double a = 3.14;
double b = 1.57;

// Объявления функций:
double f1(double x);
double f2(double x);

// Определения функций:
double f1(double x)
{
 return 4.0/(1.0+a*x+x*x);
}

// Обязательная главная функция:
int main()
{
  // Объявления переменных
  int i, n=100;
  myint A; // подставляется long long int A;
  float C = Pi; // подставляется float C = 3.1415926535897932384;
  double a, b = 1.5, h, s;
  float *Adr1;
  double *Adr2;
  double aa[10], bb[10];
  double *cc, *dd;

  int_Ar_1D_t MyStr;

  MyStr.len = 0;
  MyStr.max = 0;
  MyStr.data = NULL;
```

```
  MyStr.data = (int *)(malloc(sizeof(int)*50));
  MyStr.max = 50;

  for(i=0; i<25; i++) MyStr.data[i] = i;
  MyStr.len = 25;

  Adr1 = &C;
  Adr2 = &b;

  // Операторы:
  A = (myint) C;
  printf("A=%ld\n",A);

  a = f1(b);
  b = f2(a);
  printf("a=%lf b=%lf\n",a,b);

  s = 0;
  for (i=0; i<10; i++) {
    aa[i] = 0.1*i + 0.05; /* a_i = 0.1*i+0.05 */
    bb[i] = f1(aa[i]);    /* b_i = 4/(1+a_i*a_i) */
    s = s + bb[i];        /* s = sum_{i=0}^{i=9} b_i */
  }
  printf("s=%16.14lf Pi=%16.14lf\n",s,Pi);

  n = 10000;

// Выделение дополнительной памяти:
  cc = (double*)(malloc(sizeof(double)*n));
  dd = (double*)(malloc(sizeof(double)*n));

  h = 1.0 / n;
  s = 0;
  for (i=0; i<n; i++) {
    cc[i] = h*i + 0.5*h;  /* c_i = 0.1*i+0.05 */
    dd[i] = f1(cc[i]);    /* d_i = 4/(1+c_i*c_i) */
    s = s + dd[i];        /* s = sum_{i=0}^{i=n-1} d_i */
  }
  printf("s=%16.14lf Pi=%16.14lf\n",s,Pi);

  return 0;
}

// Определения функций:
double f2(double x)
{
 return 4.0/(1.0+b*x*x*x);
}
```

3. Типы данных.

3.1. Базовые типы данных

Базовые типы данных – целые числа, вещественные числа, символы, логические величины, специальные.

Целые типы	Вещественные типы	Символьные типы	Логический тип	Специальные

`int`
`unsigned int`
`short int`
`unsigned short int`
`long int`
`unsigned long int`
`long long int`
`unsigned long long int`
	`float`
	`double`
	`long double`
		`char`
		`unsigned char`
			`bool`	`void`
				`enum`
				`int*`
				`float*`
				`double*`
				`char*`
				`bool*`

3.2. Производные типы данных

Производные типы данных – массивы, структуры, объединения.

Массив – именованный набор данных, содержащий элементы одного из базовых типов (вектор, матрица, тензор).
Структура – именованный набор данных, содержащий элементы разных типов.
Объединение – именованный набор данных, содержащий в одних и тех же ячейках памяти вариативно элементы разных типов.

3.3. Пример использования массивов. Программа – "integral.c"

```
>vi integral.c

>cat integral.c
/*=============================================================*/
/* Программа "integral.c"                                      */
/*=============================================================*/
#include <stdio.h>
#include <stdlib.h>

double f1(double x);

int main()
{
  int i, n;
  double Pi = 3.1415926535897932384;
  double a=0, b=1.0, h, s;
  double aa[10], bb[10];
  double *cc, *dd;
  double ac;

  n = 10;
  h = (b-a)/n;

  s = 0;
  for (i=0; i<n; i++) {
    aa[i] = h*i + 0.5*h;  /* a_i = h*i+0.5*h */
    bb[i] = f1(aa[i]);    /* b_i = 4/(1+a_i*a_i) */
    s = s + h*bb[i];      /* s = sum_{i=0}^{i=9} h*b_i */
  }
  ac = s - Pi;
  printf("s=%16.14lf Pi=%16.14lf ac=%le\n",s,Pi,ac);

  n = 1000000;
  h = (b-a) / n;

  cc = (double*)(malloc(sizeof(double)*n));
  if (cc == NULL) { printf("No memory!\n"); return -1; }

  dd = (double*)(malloc(sizeof(double)*n));
  if (dd == NULL) return -2;

  s = 0;
  for (i=0; i<n; i++) {
    cc[i] = h*i + 0.5*h;  /* c_i = h*i+0.5*h */
    dd[i] = f1(cc[i]);    /* d_i = 4/(1+c_i*c_i) */
    s = s + h*dd[i];      /* s = sum_{i=0}^{i=n-1} h*d_i */
  }
  ac = s - Pi;
  printf("s=%16.14lf Pi=%16.14lf ac=%le\n",s,Pi,ac);

  free(cc);
  free(dd);

  return 0;
}

double f1(double x)
{
 return 4.0/(1.0+x*x);
}
```

Компиляция:

```
> gcc -o integral integral.c -lm

>integral
s=3.14242598500110 Pi=3.14159265358979
s=3.14159265358998 Pi=3.14159265358979
```

Интеграл от математической функции:

Добавим в код фрагмент:

```
  b = Pi/2;
  h = (b-a) / n;
  s = 0;
  for (i=0; i<n; i++) {
    cc[i] = h*i + 0.5*h; /* c_i = h*i+0.5*h */
    dd[i] = cos(cc[i]); /* d_i = 4/(1+c_i*c_i) */
    s = s + h*dd[i]; /* s = sum_{i=0}^{i=n-1} h*d_i */
  }
  printf("s=%16.14lf\n",s);
```

Компиляция:

```
> gcc -o integral integral.c -lm

>integral
s=3.14242598500110 Pi=3.14159265358979 ac=8.333314e-04
s=3.14159265358998 Pi=3.14159265358979 ac=1.825207e-13
s=1.00000000000014
```