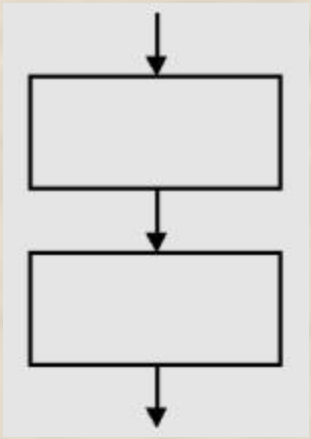
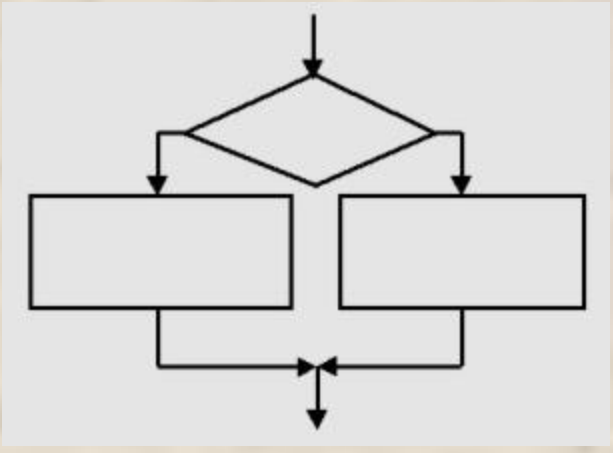
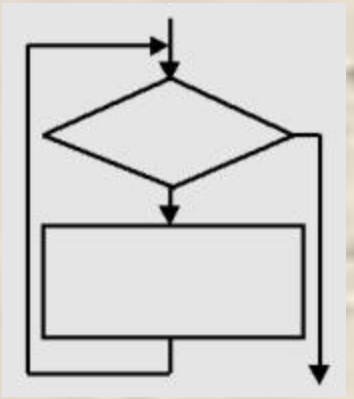
Лекция 3. Управляющие операторы. Структуры данных языка С

Рассматриваются основные операторы языка: ветвления, циклы, передача управления, а также типы и структуры данных, унаследованные из языка С: указатели, массивы, строки, перечисления, структуры и объединения.

Базовые конструкции структурного программирования







Условный оператор

if (выражение) оператор_1; [else оператор_2;]

```
   if (a<0) b = 1;                    // 1
   if (a<b && (a>d || a==0)) b++;
   else {b* = a; a = 0;}              // 2
   if (a<b){
      if (a<c) m = a;
      else m = c;}
   else {if (b<c) m = b;
         else m = c;}                 // 3
   if (a++) b++;                      // 4
if (b>a) max = b;
else max = a;          // max = (b > a) ? b : a;
```

```
if (int i = fun(t)) a -= i; else a += i;
```

Оператор **switch**

```
switch ( выражение ){
   case константное_выражение_1:
        [список_операторов_1]
   case константное_выражение_2:
        [список_операторов_2]
   ...
   case константное_выражение_n:
        [список_операторов_n]
   [default: операторы ]
}
```

Пример оператора switch

```
#include <iostream.h>
int main(){
   int a, b, res; char op; bool f = true;
   cout << "\nВведите 1й операнд : ";      cin >> a;
   cout << "\nВведите знак операции : ";  cin >> op;
   cout << "\nВведите 2й операнд : ";      cin >> b;
   switch (op){
     case '+': res = a + b; break;
     case '-': res = a - b; break;
     case '*': res = a * b; break;
     case '/': res = a / b; break;
     default : cout <<"\nНеизвестная операция"; f = false;
   }
   if (f) cout << "\nРезультат : " << res;
}
```

Оператор цикла **while**

while (выражение) оператор

```
#include <stdio.h>
int main(){
  float Xn, Xk, Dx;
  printf("Введите диапазон и шаг изм-я аргумента: ");
  scanf("%f%f%f", &Xn, &Xk, &Dx);
  printf("|   X   |   Y   |\n");
  float X = Xn;
  while (X <= Xk){
     printf("| %5.2f | %5.2f |\n", X, X*X + 1);      X
+= Dx;
  }
}
```

while (int x = 0) { /* область действия x */ }

Оператор цикла **do while**

do оператор **while** выражение;

```
#include <iostream.h>
int main(){
   char answer;
   do{
      cout << "\nКупи слоника! ";
      cin >> answer;
   }while (answer != 'y');
}
```

Пример 6 - вычисление квадратного корня

```
#include <stdio.h>
#include <math.h>
int main(){
   double X, Eps;
   double Yp, Y = 1;
   printf("Введите аргумент и точность: ");
   scanf("%lf%lf", &X, &Eps);
   do{
      Yp = Y;
      Y = (Yp + X/Yp)/2;
   }while (fabs(Y - Yp) >= Eps);
   printf("\n %lf %lf", X, Y);
}
```

$$y_n = \frac{1}{2}\left(y_{n-1} + x/y_{n-1}\right)$$

Оператор цикла **for**

for (инициализация; выражение; модификации) оператор;

```
for (int i = 1, s = 0; i<=100; i++) s += i;
```

```
#include <iostream.h>
int main(){
    int num;
    cout << "\nВведите число : "; cin >> num;
    for (int half = num / 2, div = 2; div <= half; div++)
     if (!(num % div))cout << div <<"\n";
}
```

Операторы передачи управления

- оператор безусловного перехода goto;
- оператор выхода из цикла break;
- оператор перехода к следующей итерации цикла continue;
- оператор возврата из функции return.

Пример 7 - Вычисление суммы ряда

$$\mathrm{sh}\ x = 1 + x^3/3! + x^5/5! + x^7/7! + \ldots$$

```
#include <iostream.h>
#include <math.h>
int main(){
   const int MaxIter = 500;
   double x, eps;
   cout << "\nВведите аргумент и точность: ";
   cin >> x >> eps;
   bool ok = true;
   double y = x, ch = x;
   for (int n = 0; fabs(ch) > eps; n++){
      ch *= x * x /(2 * n + 2)/(2 * n + 3);
      y += ch;
      if (n > MaxIter){ok = false; break;}
   }
   if (ok) cout << "\nЗначение функции: " << y;
   else    cout << "\nРяд расходится!";
}
```

Операция получения адреса &

Унарная операция получения адреса & применима к величинам, имеющим имя и размещенным в оперативной памяти. Таким образом, нельзя получить адрес скалярного выражения, неименованной константы или регистровой переменной.

```
int a = 5;
int* p = &a;
```

Массивы

```
#include <iostream.h>
int main(){
   const int n = 10;
   int marks[n] = {3, 4, 5, 4, 4};
   int i, sum;
   for ( i = 0, sum = 0; i<n; i++)
      sum += marks[i];
   cout << "Сумма элементов: " << sum;
}
```

```
   int a[100], b[100];
   int *pa = a;   // или int *p = &a[0];
   int *pb = b;
   for(int i = 0; i<100; i++)*pb++ = *pa++;
      // или pb[i] = pa[i];
```

```
float p[10]; *u[20];
int a[5] = {1, 2, 3};
int b[] = {1, 2, 3};
char cv[4] = { 'a', 's', 'd', 'f', 0 };        //  error
```

```
p[5]     5[p]     *(p+5)
```

Пример - сортировка выбором

```
#include <iostream.h>
int main(){
    const int n = 20;          int b[n];      int i;
    for (i = 0; i<n; i++) cin >> b[i];
    for (i = 0; i<n-1; i++){
        int imin = i;
        for (int j = i + 1; j<n; j++) if (b[j] < b[imin]) imin = j;
        int a = b[i]; b[i] = b[imin];  b[imin] = a;
    }
    for (i = 0; i<n; i++)cout << b[i] << ' ';
    return 0;
}
```

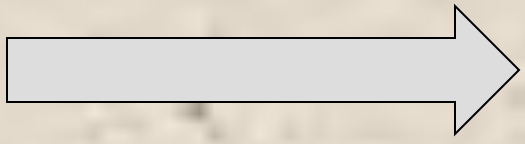
Динамические массивы

```
float *p = new float [100];
float *q = (float *) malloc(100 * sizeof(float));
```

```
delete [] p; free
        (q);
```

Многомерные массивы

int matr [6][8];



```
matr[i][j]
*(matr[i]+j)
*(*(matr+i)+j)
```

```
int mass2 [][2]={ {1, 1}, {0, 2}, {1, 0} };
int mass2 [3][2]={1, 1, 0, 2, 1, 0};
```

```
int x3d[3][5][7];
float y[4][3] = {  { 1 }, { 2 }, { 3 }, { 4 } };
// первый столбец, остальные 0
```

```
int nstr = 5;
int ** m = (int **) new int [nstr][10];
```

```
int nstr, nstb;
cout << " Введите количество строк и столбцов :";
cin >> nstr >> nstb;
int **a = new int *[nstr];          // 1
for(int i = 0; i<nstr; i++)         // 2
   a[i] = new int [nstb];           // 3
```

Строки

Строка - массив символов, заканчивающийся нуль-символом (символ с кодом, равным 0; записывается '\0').

```
char str[10] = "Vasia";    // переменная
char str[] = "Vasia";      // переменная
char *str = "Vasia”        // константа
```

Библиотека - <string.h>

Пример 1 - строки

```
#include <stdio.h>
#include <string.h>
int main(){
   char s[5], passw[] = "kuku";
/* Можно -  *passw = "kuku"; */
int i, k = 0;
   for (i = 0; !k && i<3; i++){
      printf("\nвведите пароль:\n");
      gets(s);                        if
(strstr(s,passw))k = 1;
   }
   if (k) printf("\nпароль принят");
   else printf("\nпароль не принят");
}
```

Пример 2 - строки

```
char src[10], dest[10];
for (int i = 0; i <= strlen(src); i++)
dest[i] = src[i];
```

```
char *src = new char [10];
   char *dest = new char [10], *d = dest;
   cin << src;
   while ( *src != 0) *d++ = *src++;
   *d = 0;
```

---

while (*d++ = *src++);

Типы данных, определяемые пользователем

enum

struct

union

Переименование типов (**typedef**)

typedef тип новое_имя [размерность];

```
typedef unsigned int UINT;
typedef char Msg[100];
typedef struct{
   char fio[30];
   int date, code;
   float salary;} Worker;
```

```
UINT i, j;   Msg str[10];
Worker stuff[100];
```

Перечисления (**enum**)

```
enum [ имя_типа ] { список_констант };
```

```
enum Err { ERR_READ, ERR_WRITE, ERR_CONVERT};
Err error;

switch (error){
case ERR_READ: /* оп */ break;
case ERR_WRITE:  /* оп */ break;
case ERR_CONVERT: /* оп */ break;
}
```

```
enum {two = 2, three, four, ten = 10,
eleven, fifty = 50};
```

Структуры (struct)

```
struct [ имя_типа ] {
тип_1 элемент_1;
тип_2 элемент_2;
...
тип_n элемент_n;
} [ список_описателей ];
```

```
 struct {
    char fio[30];
    int date, code;
    float salary;
}staff[100], *ps;
```

```
struct List;
   struct Link{
   List *p;
   Link *prev,
*succ;
};
struct List {
/* опр-е List */};
```

Инициализация структур

```
struct{
   char fio[30];
   int date, code;
   float salary;
}worker = {"Страусенко", 31, 215, 3400.55};
```

```
struct complex {
   float real, im;
} compl [2][3]={
{{1, 1}, {1, 1}, {1, 1}},
{{2, 2}, {2, 2}, {2, 2}}
};
```

Доступ к полям структуры

```
Worker worker, staff[100], *ps;

worker.fio = "Страусенко";
staff[8].code = 215;
ps->salary = 0.12;



struct A {int a; double x;};
struct B {A a; double x;} x[2];
x[0].a.a = 1;
x[1].x = 0.1;
```

```
   struct Options {
      bool centerX:1;
      bool centerY:1;
      unsigned int shadow:2;
      unsigned int palette:4;
   };
   union {
   unsigned char ch;
   Options bit;
   }option = {0xC4};
   cout << option.bit.palette;
option.ch &= 0xF0;
```

Ограничения объединений

- объединение может инициализироваться только значением его первого элемента;
- объединение не может содержать битовые поля;
- объединение не может содержать виртуальные методы, конструкторы, деструкторы и операцию присваивания;
- объединение не может входить в иерархию классов.