Дәріс 6. Функциялар Функцияны жариялау және анықтау, параметрлер, қайтаратын мән, көріну аймағы (scope), рекурсия.

Функциялар

1. Функция туралы түсінік

Функция — бұл белгілі бір әрекетті орындауға арналған, өз алдына жеке жазылған программалық блок.
C++ тілінде функциялар бағдарламаны модульдік және құрылымдық түрде ұйымдастыруға мүмкіндік береді. Яғни үлкен бағдарламаны бірнеше шағын, түсінікті бөліктерге бөлуге болады.

Функциялардың артықшылықтары:

- Кодты қайта пайдалану мүмкіндігі (reuse);
- Бағдарламаны құрылымдау және түсінікті ету;
- Жөндеу және тестілеуді жеңілдету;
- Күрделі есептерді шағын бөліктерге бөлу.

C++ тілінде функциялар екі түрлі болады:

1. **Кітапханалық функциялар (standard functions)** — мысалы: `sqrt()`, `pow()`, `strlen()`, `printf()`, `cin`, `cout` т.б.
2. **Пайдаланушы анықтайтын функциялар (user-defined functions)** — яғни бағдарламалаушы өзі жазған функциялар.

---

2. Функцияның құрылымы

C++ тіліндегі функция келесі негізгі бөліктерден тұрады:

```
қайтару_типі функция_атауы(параметрлер_тізімі)
{
    // функция денесі
    операторлар;
    return мән;
}
```

Мысал:

```
int add(int a, int b)
{
    int sum = a + b;
    return sum;
}
```

Бұл функция екі бүтін санды қосып, нәтижесін қайтарады.

Функцияның құрамдас бөліктері:

- **Қайтару типі (return type)** — функция қай типтегі мән қайтаратынын көрсетеді (`int`, `double`, `void`, т.б.);
- **Функция атауы (name)** — функцияны шақыру кезінде қолданылады;
- **Параметрлер тізімі (parameters)** — функцияға берілетін аргументтер;
- **Функция денесі (body)** — функцияның негізгі коды;
- **`return` операторы** — функциядан нәтижені қайтару үшін қолданылады.

---

3. Функцияны жариялау және анықтау

C++ тілінде функцияны қолданбас бұрын оны **жариялау (declaration)** қажет. Бұл компиляторға функцияның бар екенін алдын ала хабарлайды.

Функцияны жариялау (прототип)

```
int add(int, int); // функция прототипі
```

Функцияны анықтау

```
int add(int x, int y) {
    return x + y;
}
```

Функцияны шақыру

```
int main() {
    int result = add(5, 3);
    cout << "Нәтиже: " << result;
}
```

---

4. Параметрлер және аргументтер

Функция параметрлері арқылы негізгі программа мен функция арасында деректер алмасады.

Параметр түрлері:

1. **Мән арқылы беру (Pass by Value)** — аргументтің көшірмесі беріледі.
```
2. void showValue(int x) { x = 10; }
```

   Мұнда негізгі айнымалының мәні өзгермейді.

3. **Сілтеме арқылы беру (Pass by Reference)** — айнымалының өзін береді.
```
4. void modify(int &x) { x = 10; }
```

   Бұл жағдайда негізгі айнымалының мәні өзгереді.

5. **Көрсеткіш арқылы беру (Pass by Pointer)** — айнымалының адресі беріледі.

6. `void modify(int *x) { *x = 10; }`

Мысал:

```
void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}
```

---

5. Функцияның қайтару мәні

Функция бір мәнді `return` операторы арқылы қайтарады.

```
double square(double x) {
    return x * x;
}
```

Ескерту:
Егер функция ештеңе қайтармаса, `void` типі қолданылады:

```
void printHello() {
    cout << "Сәлем, әлем!";
}
```

---

6. Функцияның көріну аймағы (Scope)

C++ тілінде айнымалылардың көріну аймағы **(scope)** олардың қай жерде жарияланғанына байланысты:

- **Жергілікті айнымалы (Local variable)** — функция ішінде жарияланады және тек сол функцияда қолданылады.
- **Жалпы айнымалы (Global variable)** — функциядан тыс жарияланады және барлық функциялар қол жеткізе алады.
- **Блоктық айнымалы (Block scope)** — `{ }` ішіндегі код блогына тиесілі.

Мысал:

```
int globalVar = 5; // глобалдық айнымалы

void func() {
    int localVar = 10; // локалдық айнымалы
    cout << globalVar << " " << localVar;
}
```

---

7. Әдепкі параметрлер (Default Parameters)

C++ функциялары параметрлерге әдепкі мән тағайындауға мүмкіндік береді.

```
int multiply(int a, int b = 2) {
```

```
    return a * b;
}

int main() {
    cout << multiply(5);    // 10
    cout << multiply(5, 3); // 15
}
```

---

8. Аргументтер саны өзгермелі функциялар

Кейбір функциялар аргументтердің кез келген санын қабылдай алады (`<cstdarg>` кітапханасы арқылы).
Мысалы: `printf()`

```
#include <cstdarg>
#include <iostream>
using namespace std;

int sum(int count, ...) {
    va_list args;
    va_start(args, count);
    int total = 0;
    for (int i = 0; i < count; i++)
        total += va_arg(args, int);
    va_end(args);
    return total;
}
```

---

9. Рекурсия (Recursion)

Рекурсия — функцияның өзінің ішінен өзін қайта шақыруы.

Рекурсивті функция әрқашан екі бөлімнен тұрады:

1. **Негізгі шарт (Base case)** — рекурсия тоқтайтын нүкте;
2. **Рекурсивті шақыру (Recursive call)** — функция өзін қайта шақырады.

Мысал 1: Факториал есептеу

```
int factorial(int n) {
    if (n <= 1) return 1;  // Негізгі шарт
    else return n * factorial(n - 1); // Рекурсивті шақыру
}

int main() {
    cout << factorial(5); // 120
}
```

Мысал 2: Фибоначчи тізбегі

```
int fib(int n) {
    if (n <= 1) return n;
    return fib(n - 1) + fib(n - 2);
}
```

10. Inline функциялар

Inline функциялар — қысқа және жиі қолданылатын функцияларды орындауды жеделдету үшін қолданылады.

```
inline int square(int x) {
    return x * x;
}
```

Компилятор мұндай функцияларды шақырғанда оның денесін тікелей кодқа енгізеді.

---

11. Функцияны артық жүктеу (Function Overloading)

C++ тілінде бір атаумен бірнеше функция жазуға болады, егер олардың параметрлері әртүрлі болса.

```
int add(int a, int b) { return a + b; }
double add(double a, double b) { return a + b; }
```

Компилятор аргументтер типіне қарай тиісті функцияны таңдайды.

---

12. Функциялар және модульдік бағдарламалау

Функциялар бағдарламаны модульдерге бөлуге мүмкіндік береді. Әрбір модуль жеке файлда орналасып, басқа файлдарда пайдаланылуы мүмкін.

Мысалы:

- **mathfuncs.h**
- `int add(int a, int b);`
- `int multiply(int a, int b);`
- **mathfuncs.cpp**
- `#include "mathfuncs.h"`
- `int add(int a, int b) { return a + b; }`
- `int multiply(int a, int b) { return a * b; }`
- **main.cpp**
- `#include <iostream>`
- `#include "mathfuncs.h"`
- `using namespace std;`
-
- `int main() {`
- `    cout << add(2, 3);`
- `}`

Қорытынды

Функциялар — C++ тілінің негізгі құрылымдық элементтерінің бірі. Олар:

- бағдарламаны модульдерге бөледі;
- кодтың түсініктілігін арттырады;
- қайталануды азайтады;
- тиімді және ықшам бағдарламалар жазуға мүмкіндік береді.

Өзіндік бақылау сұрақтары

1. Функция дегеніміз не және ол не үшін қажет?
2. Функцияның құрылымы қандай бөліктерден тұрады?
3. Функцияны жариялау мен анықтау айырмашылығы неде?
4. Параметрлер мен аргументтердің айырмашылығы неде?
5. Қайтару типі қандай қызмет атқарады?
6. Рекурсия қалай жұмыс істейді?
7. Inline функциялар не үшін қолданылады?
8. Әдепкі параметрлердің маңызы қандай?
9. Артық жүктелген функциялардың ерекшелігі неде?
10. Көріну аймағы ұғымын түсіндіріңіз.