Программирование на C++

Минцифры России

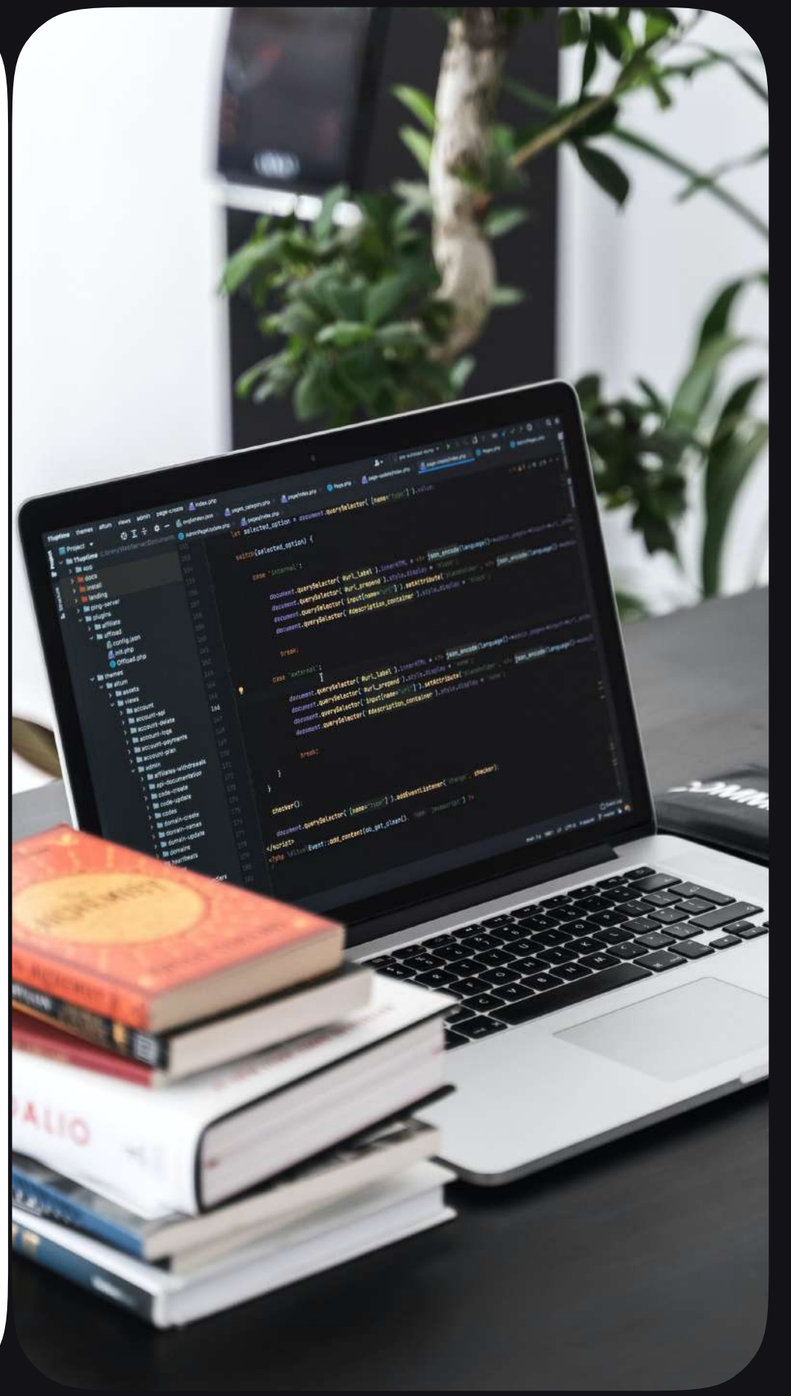
UCHi DOMA

20.35 УНИВЕРСИТЕТ

Урок 3 Модуль 3

Классы

Полезные материалы



Цели урока

- изучить абстрактные типы данных
- отработать на практике написание алгоритмов с использованием абстрактных типов данных C++

Абстрактные типы данных

Язык C++ позволяет создавать типы данных, которые ведут себя аналогично базовым типам языка Си.

Такие типы обычно называют **абстрактными типами данных**.

Для реализации абстрактных типов данных языке Си используются структуры. Но использование данных структурного типа значительно ограничено по сравнению с использованием базовых типов данных.

Абстрактные типы данных

Элементы структуры никак не защищены от случайной модификации. Любая функция может обратиться к элементу структуры. Это противоречит одному из основных принципов объектно-ориентированного программирования — инкапсуляции данных: никакие другие функции, кроме специальных функций манипуляции этим типом данных, не должны иметь доступ к элементам данных.

Абстрактные типы данных

Создадим структуру Struct для хранения даты date и множества функций для работы с переменными этого типа:

```
#include <stdio.h>
struct date
{
  int month; // месяц
  int day; // день
  int year; // год
};
void set_date(date* f, int d, int m, int y)
{
  f->day = d;
  f->month = m;
  f->year = y;
}
void print_date(date* f)
{
  printf("%d.%d.%d", f->day, f->month, f->year);
}
int main()
{
  date today;
  set_date(&today, 01, 01, 2023);
  print_date(&today);
  return 0;
}

```

Абстрактные типы данных

Функции set_date и print_date не имеют никакой явной связи с типом данных.

Для вызова любой из описанных функций требуется в качестве аргумента передать указатель на экземпляр структуры.

Абстрактные типы данных

Связь функций и данных можно установить, описав функции как члены структуры.
Эти функции могут действовать на данные, содержащие в самой структуре.

По умолчанию при объявлении структуры ее данные и функции являются общими, то есть у объектов типа структура нет ни инкапсуляции, ни защиты данных:

```
1   #include <stdio.h>
2   struct date
3   {
4     int month; // месяц
5     int day; // день
6     int year; // год
7     void set_date(int d, int m, int y)
8     {
9       day = d; month = m; year = y;
10    }
11    void print_date(void);
12  };
13  void date::print_date(void)
14  {
15    printf("%d.%d.%d", day, month, year);
16  }
17  int main()
18  {
19    date today;
20    today.set_date(2, 4, 2014);
21    today.print_date();
22    getchar();
23    return 0;
24  }
```

Функции-члены и данные-члены

Функции, описанные в теле абстрактного типа данных, представляют собой **функции-члены** или **методы** и могут вызываться только для специальной переменной соответствующего типа с использованием стандартного синтаксиса для доступа к **данным-членам** или **полям** структуры.

Функции-члены и данные-члены

Определение функций-членов может осуществляться двумя способами:

- описание функции непосредственно при описании структуры
- описание функции вне структуры

Функции-члены, которые определены внутри структуры, являются неявно встроенными

В функции-члене имена членов могут использоваться без явной ссылки на объект. В этом случае имя относится к члену того объекта, для которого функция была вызвана.

тип АТД::имя(список аргументов)
{
тело функции-члена;
}

- **тип** — тип возвращаемого значения функции-члена
- **АТД** — имя абстрактного типа данных (имя структуры или класса)
- **имя** — имя функции-члена

```
void date::print_date(void)
{
    printf("%d.%d.%d",day, month, year);
}
```

Права доступа

Концепция структуры в языке C++ (в отличие от Си) позволяет членам структуры быть общими, частными или защищенными:

- **public** — общие
- **private** — частные
- **protected** — защищенные

Права доступа

Использование ключевого слова **protected** связано с понятием **наследования.**

Использование ключевого слова **private** ограничивает доступ к членам, которые следуют за этой конструкцией. Члены **private** могут использоваться только несколькими категориями функций, в привилегии которых входит доступ к этим членам. В основном это функции-члены той же структуры.

Ключевое слово **public** образует интерфейс к объекту структуры.

Права доступа

Стандартным является размещение член-данных в частной области (**private**), а части функций-членов — в общей части (**public**) абстрактного типа данных. В этом случае закрытая (**private**) часть определяет данные объекта и служебные функции, а функции-члены общей части реализуют методы работы с объектом.

Права доступа

Изменим структуру **date** так, чтобы скрыть представление данных (инкапсуляция данных):

```
struct date
{
  private:
    int month, day, year;
  public:
    void set(int, int, int);
    void print();
};
```

Закрепление

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

struct Adder {
    public:
      void addNum(int number) {
          total += number;
      }
      int getTotal() {
        return total;
      };
    private:
      int total;
};
int main() {
    Adder a;

    a.addNum(10);
    a.addNum(20);
    a.addNum(30);

    cout << "Total " << a.getTotal() <<endl;
    return 0;
}
```

Что будет выведено на экран
в результате работы программы?

Закрепление

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

struct Adder {
    public:
      void addNum(int number) {
          total += number;
      }
      int getTotal() {
        return total;
      };
    private:
      int total;
};
int main() {
    Adder a;

    a.addNum(10);
    a.addNum(20);
    a.addNum(30);

    cout << "Total " << a.getTotal() <<endl;
    return 0;
}
```

Что будет выведено на экран
в результате работы программы?

Результат работы программы:

```
Total 60
```