ДАТЫ В C++

ОАиП

Аверина И.Н.

Работа с датами в C++

Стандартная библиотека C ++ не имеет встроенного типа даты.

C ++ наследует структуры и функции для обработки дат и времени из C.

Чтобы получить доступ к функциям и структурам, связанным с датой и временем, используют

- библиотеку (класс) **<ctime>** .
- библиотеку **boost**

Или используют переменные типа структура и пользовательские алгоритмы

Класс <ctime>

Существует четыре типа времени:

clock_t, time_t, size_t

и структура **tm**

`struct tm`

Member	Type	Meaning	Range
tm_sec	int	seconds after the minute	0-60*
tm_min	int	minutes after the hour	0-59
tm_hour	int	hours since midnight	0-23
tm_mday	int	day of the month	1-31
tm_mon	int	months since January	0-11
tm_year	int	years since 1900	
tm_wday	int	days since Sunday	0-6
tm_yday	int	days since January 1	0-365
tm_isdst	int	Daylight Saving Time flag	

<ctime> (time.h)

- ***functions:***
  - asctime
  - clock
  - ctime
  - difftime
  - gmtime
  - localtime
  - mktime
  - strftime
  - time
- ***macros:***
  - CLOCKS_PER_SEC
  - NULL
- ***types:***
  - clock_t
  - size_t
  - time_t
  - struct tm

Библиотека C++ Boost

Набор библиотек даты и времени, основанный на общих концепциях программирования.

- Библиотека богата терминологией и проблемами.
- Библиотека поддерживает 3 основных временных типа:

Точка времени - указатель местоположения на оси времени

Продолжительность времени - отрезок времени, не привязанный к какой-либо точке времени

Интервал времени - продолжительность времени, привязанная к определенной точке во временном континууме. Также известен как период времени.

Вводим дату с клавиатуры как строку из 10 символов,

ДАТА – это символьный массив из 10 элементов

_ _ /_ _ /_ _ _ _

s[0] s[1] **s[2]** s[3] s[4] **s[5]** s[6] s[7] s[8] s[9]

Каждый символ цифры имеет соответствующий ему код в базовой таблице кодировки:

Символ цифры	Десятичн код
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

```
char s[10];
cout<<"Enter date (dd/mm/yyyy):";
cin>>s;
//символ '1' преобразовывается в число 49
//числовой код символа 0   (int)'0'= 48
//их разница даст цифру 1
Sd= ((int)s[0]-48)*10 + ((int)s[1]-48);
Sm= ((int)s[3]-48)*10 + ((int)s[4]-48);
Sy= ((int)s[6]-48)*1000+((int)s[7]-48)*100 +
    ((int)s[8]-48)*10 + ((int)s[9]-48);
cout<<"day = "<<Sd<<"  month= "<<Sm
    <<"Year= "<<Sy<<endl;
```

```
Enter date (dd/mm/yyyy):14/07/1967
day = 14  month= 7  Year= 1967
```

Вычислить возраст на текущую дату

(вычисление разницы дат)

Простой способ, но очень приблизительный результат

26/10/1995 - 29/03/2021

"26/10/1995" = 1995*365+10*30+26 = 728501 дней

"29/03/2021" = 2021*365+ 3*30 +29 = 737784 дней

737784 - 728501 = 9283 (дней между датами)

9283/365=25,43 → 25 полных лет

43*12/100 =5,16 →5 месяцев

16*30/100=4,8 →4 дня

вычисление разницы дат DD.MM.GGGG и dd.mm.gggg

```
int d, m, g;
int D, M, G;

cout << "Дата текущая:  1 -  29 03 2021" << endl;
cout << "Дата рождения: 2 -  26 10 1995" << endl;
cout << endl;
cout << "Введите 1-ю дату: "; cin >> D >> M >> G;
cout << "Введите 2-ю дату: "; cin >> d >> m >> g;
cout << " " << endl;
```

```
if (d>D)
  {  D=D+31;
     d=D-d;
     M=M-1; }
else // (d<D)
  {    d=D-d;}
if (m>M)
  {  M=M+12;
     m=M-m;
     G=G-1; }
else
  {  m=M-m; }
g=G-g;
```

```
Введите 1-ю дату: 29 03 2021
Введите 2-ю дату: 26 10 1995

Возраст 25 лет  5 месяцев  3 дней
```

```
Введите 1-ю дату: 29 03 2021
Введите 2-ю дату: 06 12 2014

Возраст 6 лет  3 месяцев  23 дней
```

```
cout << "Возраст " << g << " лет  "
     << m << " месяцев  " << d << " дней" << endl;

_getch();
```

Работа с датами в С ++, используя структуры

```
struct Date
{
    int day,month,year;
};

void input_Date(Date &d){
    cout<<"Day - ";
    cin>>d.day;
    cout<<"\nMonth - ";
    cin>>d.month;
    cout<<"Year - ";
    cin>>d.year;
}

void output_Date(Date &d){
    cout<<d.day<<"."<<d.month<<"."<<d.year;
}
```

Составить функцию, которая возвращает ответ на вопрос
«событие произошло позже указанной даты?».

Параметры функции:

указатель на структурную переменную с полями целого типа **day, month, year** (для указания даты, когда произошло событие), указатель на С-строку (дата в формате “**YYYY.MM.DD**”, относительно которой происходит сравнение).

Результат функции:

true, *если ответ утвердительный;*
false, в противном случае.

```
struct date{
    int day, month, year;
};
```

Функция сравнения дат

```
bool Ex(date d, char* s)
{
    int sy = 0, sm = 0, sd = 0, i;
    for(i = 3; i >= 0; i--)
        sy = sy * 10 + ((int)s[i] - (int)'0');
    // …

    bool f = false;
    if (sy < d.year) f = true;
    else if (sy == d.year && sm < d.month) f = true;
    else if (sy == d.year && sm == d.month && sd < d.day) f = true;
    return f;
}
```

Главный модуль

```
int main() {
        char Date[15];
        cout << "Enter date: "; cin.getline(Date, 11);

        date D = { 4,10,2018 };

        cout << boolalpha;
        cout << Date << " < " << D.year << ".";
        cout << D.month << "." << D.day << " = ";
        cout << Ex(D, Date) << endl;
        return 0;
}
```

СТРУКТУРЫ В С++

ОАиП

Аверина И.Н.

Структуры

Структуры – составной тип данных, который под одним именем объединяет несколько переменных разных типов

Поскольку структуры определяются программистом, то вначале мы должны сообщить компилятору, как она вообще будет выглядеть. Для этого используется **ключевое слово struct**

Переменные, которые являются частью структуры, называются **членами структуры** (или ***«полями структуры»***).

1. Описание состава структуры

- это определение шаблона, который описывает данные, входящие в состав структуры и по которому впоследствии создаются переменные структуры

```
struct имя {
   тип поле1;
   тип поле2;
   . . .
   тип полеN;
} [список переменных];
```

Предупреждение:

Одна из самых простых ошибок в C++ — забыть ; в конце объявления структуры. Это приведет к ошибке компиляции в следующей строке кода.

2. Объявление переменной структурного типа

переменная структуры – *объект, который имеет* ***имя*** (имя переменной структуры) и ***поля***, тип и название которых определяются описанием структуры

имя_структуры имя_переменной_структуры;

- *раздельный* способ объявления

```
struct tovar {
        char name[20];
        int kod[7];
        float price;
};
tovar S1, S2;
```

- *совместный* способ объявления

```
struct tovar {
        char name[20];
        int kod[7];
        float price;
} S1, S2;
```

3. Доступ к элементам структуры

доступ к полям переменной структуры осуществляется через точечный синтаксис

имя_переменной_структуры.имя_поля

①
```
struct tovar {
        char name[20];
        int kod[7];
        float price;
};
```
②
```
tovar S1, S2;
```

③
```
gets(S1.name);
int i;
for (i = 0; i < 7; i++)
        cin >> S1.kod[i];
S1.price = 2.75;
```

Вложенные структуры

```
struct date { int day, month, year; } ;
struct Student {
  int  studID;
  char FIO[50];
  date DateBirth;
  } S;
cin>>S.studID; cin.ignore();
cin.getline(S.FIO,50);   //строка с пробелами
cin>>S.DateBirth.day;
cin>>S.DateBirth.month;
cin>>S.DateBirth.year;
```

```cpp
#include <stdio.h>
#include <iostream>
using namespace std;

struct date { int day, month, year; } ;
struct Student {
    int  studID;
    char FIO[50];
    date DateBirth;
    } S;
int main()
{
cout<<"enter ID "; cin>>S.studID; cin.ignore();
cout<<"enter FIO ";cin.getline(S.FIO,50);
cout<<"enter day   of Birth ";cin>>S.DateBirth.day;
cout<<"enter month of Birth ";cin>>S.DateBirth.month;
cout<<"enter year  of Birth ";cin>>S.DateBirth.year;
return 0;
}
```

4. Указатели на структуру

имя_структ_типа *имя_указателя_на_структуру

Чтобы **получить значение элемента структуры** через указатель на эту структуру, т.е. если в переменной хранится адрес (указатель на структурную переменную), то доступ к полям структуры осуществляется через знак **->**

имя_указателя_на_структуру -> имя_поля

или

(* имя_указателя_на_структуру).имя_поля

Чтобы **получить адрес структурной переменной** :

имя_указателя_на_структуру = & имя_структ_типа

5. Статический массив структур

- определить **структурный тип**;
- **объявить массив** переменных определенного **структурного типа**

```
struct  point  {
            float  x, y;
};
const  int n = 5;
point masP[n];

for(int i = 0; i < 5; i++){
            cin >> masP[i].x;
            cin >> masP[i].y;
}
```

6. Статический массив структур с указателями

- определить **структурный тип**;
- **объявить массив** переменных структурного типа;
- **объявить указатель на массив** структурного типа;
- **инициировать указатель** на начало массива структур
- **обработать** массив

```
struct point
    {
    int x,y;
    };

const int n=5;
point masP[n];

point *ptr_masP;
ptr_masP = &masP[0];

for(int i = 0; i < n; i++){
    cin>>(ptr_masP+i)->x;
    cin>>(ptr_masP+i)->y;}
```

При выполнении операций с указателями следует помнить:

– изменение указателя **ptr** на некоторую величину **i** означает не смещение на **i** байтов, а **смещение на i объектов**, каждый из которых занимает столько байтов, сколько определено для типа, на который указывает указатель

– унарные операции **взятия адреса &** и **разыменования *** имеют более высокий приоритет, чем арифметические операции

– унарные операции *, ++, -- имеют одинаковый приоритет и **при размещении рядом** выполняются **справа налево**

```cpp
#include <stdio.h>
#include <iostream>
using namespace std;

struct point
    {
    int x,y;
    };

int main() {
 point masP[3];
 point *ptr_masP;
 ptr_masP = &masP[0];
 int i;
for(i = 0; i < 3; i++){
    cout<<"enter x-"<<i<<"  ";cin>>(ptr_masP+i)->x;
    cout<<"enter y-"<<i<<"  ";cin>>(ptr_masP+i)->y;}
for(i = 0; i < 3; i++)
cout<<i<<". x= "<<(ptr_masP+i)->x<<" y= "<<(ptr_masP+i)->y<<endl;

return 0;
}
```

```
enter x-0  1
enter y-0  2
enter x-1  2
enter y-1  5
enter x-2  3
enter y-2  9
0. x= 1 y= 2
1. x= 2 y= 5
2. x= 3 y= 9
```

7. Динамический массив структур

Если заранее не известно количество записей в массиве структур, то память под него выделяют **динамически**:

N - количество элементов в массиве структур

C syntax:

```
ArrayName =
        (struct_name *) malloc (N*sizeof (struct_name));
```

C++ syntax:

```
ArrayName = new struct_name [N];
```

Пример из текста лекции

«Динамическое выделение памяти для структур» (стандарт С)

```c
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <conio.h>

struct book
{   char title[15];
    char author[15];
    int cost;
};
```

```c
struct book *lib;

lib = (struct book*)malloc(3 * sizeof(struct book));

for (int i = 0; i<3; i++)
{
  printf("Введите название %d книги : ", i + 1);
  gets((lib + i)->title);  //вместо gets_s
  printf("Введите автора %d книги : ", i + 1);
  gets ((lib + i)->author);
  printf("Введите цену %d книги : ", i + 1);
  scanf("%d", &(lib + i)->cost); // вместо scanf_s
  getchar();
}

for (int i = 0; i<3; i++)
{
  printf("\n %d. %s ", i + 1, (lib + i)->author);
  printf("%s %d", (lib + i)->title, (lib + i)->cost);
}
```

Динамическое выделение памяти для массива в C++

- Динамическое выделение памяти для массивов позволяет задавать размер массива во время выделения.
- Однако C++ не предоставляет встроенный способ изменения размера массива, который уже был выделен (в стандарте C есть **realloc**)

Но и это ограничение можно обойти:

1) динамически выделив память под новый массив,
2) скопировав в новый массив все элементы из старого массива,
3) затем удалив старый массив.

Пример из текста лекции

«Динамическое выделение памяти для структур» (стандарт C++)

```
1  #include <iostream>
2  using namespace std;
3
4  struct book
5  {  char title[15];
6     char author[15];
7     int cost;
8  };
```

```
11    struct book *lib;      //указатель на структуру
12    lib = new book [1];    //память для 1-ой записи

13    for (int i = 0; i <3; i++)    // i -- счетчик записей
14    {
15      cout<<"Введите название "<<i+1<<" книги : ";
16      cin.getline((lib + i)->title,15);
17      cout<<"Введите автора "<<i + 1<<" книги : ";
18      cin.getline((lib + i)->author,15);
19      cout<<"Введите цену "<<i+1<<" книги : ";
20      cin>>(lib + i)->cost; cin.ignore();
```

Пример из текста лекции
«Динамическое выделение памяти для структур» (стандарт C++)

```
21      book *tempLib = new book[i+2]; // память +для след.записи
22      for(int k=0; k<=i;k++)
23          tempLib[k]=lib[k]; // переписали все в новое место
24      delete [] lib;          // очистили старую память
25      lib = tempLib;   //записываем в этот указатель новый адрес
26  }
```

```
27      for (i = 0; i<3; i++)
28      { cout<<i+1<<". "<<(lib + i)->author<<"\t"
29                       <<(lib + i)->title<<"\t"
30                       <<(lib + i)->cost<<endl;  }
```

Пример2. Отобрать из массива структур записи по условию

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

struct MyStruct
        {
        int par;
        };

int main()
{
        ..              ..
```

```
//исходный массив структур
MyStruct   BaseStruct[5];
//заполняем исходный массив
int i;
for (i=0; i<5; i++)
{       cout<<"Enter element "<<i << " = ";
        cin>>BaseStruct[i].par;  }
```

```
//массив указателей  на записи,
//которые  содержат адреса записей
MyStruct   **NewStruct = NULL;
```

```
int NewStructCount=0;                    //считаем число записей в новом массиве,
for (i=0; i<5; i++)                      // условие отбора может быть ЛЮБОЕ
        if (BaseStruct[i].par >20 && BaseStruct[i].par <50)
            NewStructCount++;
```

```
//память для массива указателей
NewStruct  = new MyStruct *[NewStructCount];

//заполняем новый массив адресами отобранных записей
int k=0;
for (i=0; i<5; i++)
        if (BaseStruct[i].par>20  && BaseStruct[i].par<50)
            NewStruct[k++]  = &BaseStruct[i];

 //выводим на печать новый массив
 for (k=0; k<NewStructCount; k++)
    cout<<"NewStruct[  "<<k<<"].par = "<<NewStruct[k]->par<<endl;

 delete[] NewStruct;        // освобождаем память
```

Пример 3.

Функция выделения памяти при вводе новой записи в динамический массив структур

Функция возвращает указатель на структуру **MyStruct**

Функция принимает 2 параметра – указатель на структуру и количество структур **amount**

```
1.    MyStruct  *AddStruct(MyStruct  *Obj, const int amount)
2.    {
3.    if (amount == 0)
4.    {
5.    // выделение памяти для первой структуры
6.    Obj = new MyStruct[amount + 1];
7.    }
```

Если функция вызывается не в первый раз

```
8.      else
9.      {
10.     // выделение памяти для следующих структур
11.     MyStruct *tempObj = new MyStruct [amount + 1];
12.     // копируем записи во временный объект
13.     for (int i = 0; i < amount; i++)
14.         tempObj[i] = Obj[i];
15.     delete [] Obj;      // освобождаем старую память
16.     Obj = tempObj;      // даем указатель на новую
17.     }
18.     return Obj;
19.     }
```

Основной модуль для Примера 3.

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

struct MyStruct
        {
        int par;
        };

// прототип функции для выделения памяти для  массива структур
MyStruct *AddStruct(MyStruct  *Obj, const int amount);

1.      int main()
2.      {
3.      MyStruct *OurRecords = 0;
4.      int recordAmount = 0;
5.      int YesOrNot = 0; //  продолжить или остановить ввод данных
6.      do
7.      {
8.      OurRecords = AddStruct(OurRecords, recordAmount);
9.      /* блок заполнения новой структуры */
```

Продолжение основного модуля для Примера 3.

```
10.     recordAmount++;

11.     cout << "Продолжить ввод данных (1 - да, 0 - нет): ";
12.     cin >> YesOrNot;
13.     cin.get();
14.     } while (YesOrNot != 0);

15.     /* блок вывода всех записей */

16.     delete[] OurRecords;
17.     return 0;
18.     }
```

пример Постановки задачи

Сформировать массив структур комбинированного типа, каждый элемент которого содержит следующие поля:

- структура: баллы по экзаменам 1, 2 и 3;
- структура: ФИО студента, курс, оценки за сессию.

Определить вспомогательные функции, реализующие

- ввод данных в массив;
- вывод данных из массива;
- сортировку по полю ФИО;
- выбор данных по критерию «студенты определенного курса, которые сдали экзамены со средним баллом не меньше заданного значения»

Структуры

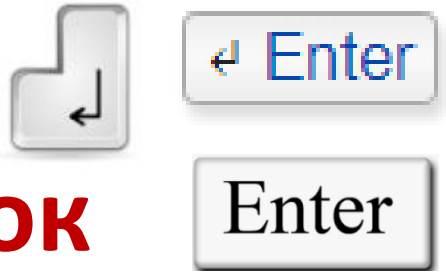
```
// описание структуры результатов по трем экзаменам
struct exam {
        int b1, b2, b3;
};

// описание структуры информации о студенте
struct stud {
        char FIO[20];
        int year;
        exam zn;
        float avg;
};
```

Прототипы функций

```
// процедура для ввода данных в массив структур
void Input_Data(stud* arr, int N);
// процедура для вывода шапки (строки заголовков) таблицы
void Top_List();
// процедура для вывода данных из массива структур
void Output_Data(stud* arr, int N);
// процедура для вывода данных элемента структуры
void Output_Elem(stud el);
// процедура для сортировки массива по полю ФИО
void Sort_FIO(stud* arr, int N);
// процедура для вывода данных из массива структур по условию
void Output_Data(stud* arr, int N, int y, float z);
```

Ввод данных в массив структур

```
void Input_Data(stud* arr, int N)
{
	for(int i = 0; i < N; i++)
	{
		cout << "Enter element " << i + 1 << endl;
		cin.ignore();
		cout << "FIO: "; cin.getline(arr[i].FIO, 20);
		cout << "year of study: "; cin >> arr[i].year;
		arr[i].avg = 0;
		cout << "exam 1: "; cin >> arr[i].zn.b1; arr[i].avg += arr[i].zn.b1;
		cout << "exam 2: "; cin >> arr[i].zn.b2; arr[i].avg += arr[i].zn.b2;
		cout << "exam 3: "; cin >> arr[i].zn.b3; arr[i].avg += arr[i].zn.b3;
		arr[i].avg /= 3.;
		cout << endl;
	}
}
```



Проблемы при вводе строк

- **cin** рассматривает **пробел** как разделитель строк
- Нажатие клавиши **Enter** дает два непечатаемых символа
  - '**\n**' перевод на новую строку (*new line, line feed, LF*) (**код #10**)
  - '**\r**'возврат каретки (курсора) в начало новой строки (*carriage return, CR*) (**код #13**)
- После **cin** или **get()** в буфере остается символ начала новой строки (**код #13**)
- Следующий **get** или его разновидность считывает этот символ как разрыв строки и заканчивает свою работу
- Поэтому, если используем разновидности **get** два раза подряд, надо очищать между ними буфер потока -- **cin.ignore()** или **cin.get()**
- **getline()** не оставляет в буфере символ #13

Форматированный вывод шапки

```
void Top_List()
{
        cout.width(4); cout << right << "#";
        cout.width(22); cout << right << "FIO";
        cout.width(5); cout << right << "year";
        cout.width(4); cout << right << "ex1";
        cout.width(4); cout << right << "ex2";
        cout.width(4); cout << right << "ex3";
        cout.width(7); cout << right << "avg";
        cout << endl;
}
```

Форматированный вывод записи

```
void Output_Elem(stud el)
{
        cout.width(22); cout << right << el.FIO;
        cout.width(5); cout << right << el.year;
        cout.width(4); cout << right << el.zn.b1;
        cout.width(4); cout << right << el.zn.b2;
        cout.width(4); cout << right << el.zn.b3;
        cout.width(7); cout.precision(2);
        cout << right << el.avg;
        cout << endl;
}
```

Форматированный вывод записей с шапкой

```
void Output_Data(stud* arr, int N)
{
        cout << "Array of data" << endl;
        cout << fixed;
        Top_List();
        for(int i = 0; i < N; i++)
        {
                cout.width(4); cout << right << i + 1;
                Output_Elem(arr[i]);
        }
}
```

Сортировка по ФИО с помощью функции strcmp( )

```
void Sort_FIO(stud* arr, int N)
{
	int i, j, im;
	stud t;
	for (i = 0; i < N - 1; i++)
	{
		im = i;
		for (j = i + 1; j < N; j++)
			if (strcmp(arr[im].FIO, arr[j].FIO) > 0) im = j;
		t = arr[i];
		arr[i] = arr[im];
		arr[im] = t;
	}
}
```

Выборка по условию

выбор данных по критерию «студенты определенного курса, которые сдали экзамены со средним баллом не меньше заданного значения»

```
void Output_Data(stud* arr, int N, int y, float z)
{
	cout << "Array of data" << endl;
	cout << fixed;
	Top_List();
	for (int i = 0; i < N; i++)
	{
		if (arr[i].year == y && arr[i].avg >= z)
		{
			cout.width(4); cout << right << i + 1;
			Output_Elem(arr[i]);
		}
	}
}
```

Основной модуль

```
int main()
{
        int N; cout << "Enter count of data: "; cin >> N;
        stud* masS = new stud[N];
        Input_Data(masS, N);
        Output_Data(masS, N);

        Sort_FIO(masS, N);
        Output_Data(masS, N);

        int y; cout << "Enter year = "; cin >> y;
        float z; cout << "Enter value = "; cin >> z;
        Output_Data(masS, N,y,z);

        delete[] masS; masS = NULL;
        return 0;
}
```

Пример форматированного вывода

```
int n = 0, r = 69, i;  // r - количество ---
for (i = 0; i < r; i++)cout << "-"; cout << endl;
cout << fixed; cout.setf(ios::right);
cout.width(4); cout << "#";
cout.width(10); cout << "Flight";
cout.width(15); cout << "Date";
cout.width(5); cout << "ID";
cout.width(25); cout << "Name";
cout.width(10); cout << "Weight"<<endl;
for (i = 0; i < r; i++)cout << "-"; cout << endl; //печать -------
```

```
---------------------------------------------------------------------
   #    Flight          Date   ID                      Name    Weight
---------------------------------------------------------------------
   1     11-AA    10/03/2021   11          Сидоров Семен      15.50
   2     11-AA    10/03/2021   12          Аверина Ирина      18.00
   3     22-SS    20/03/2021   13            Петров Петр      10.00
   4     22-SS    20/03/2021   14       Васильев Василий       8.80
   5     22-SS    20/03/2021   15          Duke Emmanuel      28.00
---------------------------------------------------------------------
```

```
cout.width(4); cout << right <<n;
cout.width(10); cout << right <<Passenger.flightID;
cout.width(15); cout << right <<Passenger.flightDate;
cout.width(5); cout << right <<Passenger.pasID;
cout.width(25); cout << right <<Passenger.name;
cout.width(10); cout.precision(2);
cout << right <<Passenger.lauggageWeight << endl;
```