acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . OOP-structC :

- Самое начало (язык С)
  - Структуры в языке С:
    - Пример реализации функций на основе структуры:
    - Задачи
    - Что изменилось.

Самое начало (язык С)

Допустим, Вас посетила ИДЕЯ. Или она посетила преподавателя или какое-то другое начальство. И теперь Вы хотите (или придется) что-то "запрограммировать". Очевидно, что сначало это что-то в Вашем представлении выглядит очень неясно. Непонятны не только средства, которыми можно пользоваться, но и самые основные вещи: что хотим получить и что имеем вначале.

Очевидно, что эту ИДЕЮ нужно предварительно превратить в четкую и ясную постановку задачи, учитывая возможности, которые программист (и не только) может иметь хотя бы теоретически.

Чтобы решить задачу, нужно:

- иметь постановку задачи (то есть понять, что же есть исходного и что нужно получить);
- понять какие инструменты нужны для решения;
- иметь эти инструменты для решения задачи (если нет инстументов - создать);
- уметь применять эти инструменты так, чтобы все-таки получить желаемое.

Поэтому начнем рассматривать самую простую ИДЕЮ: создать простые программные инструменты для решения "маленькой задачи".

Маленькая задача.

Имеются часы, на циферблате которых отображаются: часы (0 ≤ целое число ≤ 23), минуты (0 ≤ целое число ≤ 59), секунды (0≤целое число≤59).

Необходимо иметь возможность:

- устанавливать значения всех параметров часов;
- печатать значения параметорв часов в "привычном" формате: **часы:минуты:секунды;**
- считывать значения параметров в "привычном" формате: **часы:минуты:секунды;**
- определять новые значения параметров часов через заданный промежуток времени.

Каждую задачу можно решить различными способами.

Структуры в языке С:

Попробуем решить задачу "традиционным" способом. Для этого необходимо описать и реализовать набор соответсвующих функций. Чтобы упростить передачу параметров (их на часах целых три), стоит описать некоторуюю структуру. Назовем ее **Time**.

```
// Структура, описывающая время (часы, минуты, секунды)
typedef struct{
  int h;    // часы
  int min;  // минуты
  int sec;  // секунды
}Time;
```

Эта структура - новый тип данных. Новое понятие, созданное из уже имеющихся встроенных типов в языке С. Теперь каждая переменная типа **Time** будет иметь три собственных поля (**h, min, sec**).

Пример объявления таких переменных и обращения к полям.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct{
  int h;    // часы
  int min;  // минуты
  int sec;  // секунды
}Time;
```

```
int main(){
  Time a,b; //объявление переменных типа Time

// поля переменных a и b не инициализируются (там компьютерный мусор)
  a.h = 12;     // присваивание значения полю h переменной a
  a.min = 45;   // присваивание значения полю min переменной a
  a.sec = 0;    // присваивание значения полю sec переменной a

// присваивание значения переменной a к переменной b
  b = a;

// печать значений переменной b:
  printf("%d:%d:%d\n", b.h, b.min, b.sec);
  return 0;

}
```

Пример объявления динамических переменных и обращения к полям через указатели на переменную.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct{
  int h;    // часы
  int min;  // минуты
  int sec;  // секунды
}Time;

int main(){
  Time *pa; //объявление указателя на переменную типа Time
   Time *pb; //объявление указателя на переменную типа Time


// Выделение динамической памяти под переменную типа Time
//  с преобразованием возвращаемого указателя к типу Time*

  pa = (Time*)malloc( sizeof(Time) );

// поля переменных a и b не инициализируются (там компьютерный мусор)
  pa->h = 12;     // присваивание значения полю h переменной a
  pa->min = 45;   // присваивание значения полю min переменной a
  pa->sec = 0;    // присваивание значения полю sec переменной a

// Теперь оба указателя "смотрят" на одну и ту же область памяти
  pb = pa;

// печать значений переменной b:
  printf("%d:%d:%d\n", pb->h, pb->min, pb->sec);

// Получение значений полей с консоли. &(b->h) - адрес поля h.

   scanf("%d%d%d",&(pb->h), &(pb->min), &(pb->sec));

 // печать значений переменной b:
  printf("%d:%d:%d\n", pa->h, pa->min, pa->sec);
// Освобождаем динамическую память
// Теперь она доступна для других данных
  free(pa);
  return 0;

}
```

Теперь можно приступать к решению задачи с часами. Заметим, что в постановке задачи требуются только инструменты для работы с часами.

Значит, нужно реализовать эти инструменты и ПРОВЕРИТЬ их работоспособность.

Пример реализации функций на основе структуры:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Структура, описывающая время (часы, минуты, секунды)
typedef struct{
```

```
  int h,min,sec;
}Time;

// Присваивание значений времени.
// В качестве параметра передается указатель на переменную типа Time
void setTime(int _h, int _min, int _sec, Time* tmp){
// присваивание значений полям
       tmp->h =_h;
       tmp->min = _min;
       tmp->sec = _sec;
};

// Печать значений полей переменной типа Time
// Переменная типа Time задается как параметр функции
// и становится локальной переменной этотй функции
void prTime(Time z){
    printf("%.2d:%.2d:%.2d\n",z.h,z.min,z.sec);
};

// Сложение двух времен (отображаются на 24-часовом циферблате)
Time add(Time t1, Time t2){
// прежде чем вернуть переменную, ее нужно иметь
     Time tmp;

     int pr, sm;

     sm = t1.sec + t2.sec;
     pr = sm / 60;
     tmp.sec = sm % 60;
     sm = t1.min + t2.min + pr;
     tmp.min = sm % 60;
     pr = sm / 60;
     sm = t1.h + t2.h + pr;
     tmp.h = sm % 24;

// возвращение переменной
     return tmp;
};

// Чтение значений полей, записанных в привычном формате
// можно локальную переменную a типа Time использовать в вычислении (это копия)
Time readTime(Time a){
    int a1,a2,a3;
    scanf("%d:%d:%d",&a1,&a2,&a3);
    setTime(a1,a2,a3,&a);
    return a;
}


// Тестирование функций:
int main(){

// Объявление переменных типа Time
    Time a,b,c;

// Установка значений переменной a
    setTime(12,40,55,&a);

// Чтение значений в переменную b
    b = readTime(b);

// Печать значений переменных a и b
    prTime(a);
    prTime(b);

// Сложить две переменные как время
    c = add(a,b);
    prTime(c);
    return 0;
}
```

Задачи

1. для типа **Time** (задача выше) реализовать функцию

```
// вычитание t1 из t2
Time sub(Time t1, Time t2);
```

2. Дробь. Рациональные числа (дроби) представлены как <знак><целая часть>(<числитель>|0/<знаменатель>)

```
/*
 Описание структуры дробь
*/
typedef struct Dr{
  char sign;    // знак дроби
  int int_part; // целая часть
  int num;      // числитель
  int denum;    // знаменатель
} Drob;
```

Написать функции:

```
/*
Считать дробь с консоли.
формат ввода <знак><целая часть>(<числитель>|0/<знаменатель>)
Пример: -12(4/7)
*/
void inputDrob(Drob *a);

/*
Печать дроби в фомате &lt;знак&gt;&lt;целая часть&gt;(&lt;числитель&gt;/&lt;знаменатель&gt;).
Пример: -0(3/5)
*/
void printDrob(Drob a);
/*
 Сложение дробей. Дробь после сложения должна быть в несократимом виде
*/
Drob sum(Drob a, Drob b);

/*
 Вычитание дробей. Дробь после вычитания должна быть в несократимом виде
*/
Drob diff(Drob a, Drob b);
/*
Умножение дробей. Дробь после умножения должна быть в несократимом виде
*/
Drob mult(Drob a, Drob b);
/*
 Деление дробей. Дробь после деления должна быть в несократимом виде
*/
Drob division(Drob a, Drob b);
/*
Печать как десятичную с периодом. <целая часть>.(дробная периодическая часть).
Если периодическая часть =0, то печатается (0)
*/
void printAsDecimal(Drob a);
```

3. Точка на плоскости задается координатами **x** и **y**. Описать структуру **Coord** и реализовать функцию изменения координат.

```
typedef struct{
   float x,y;
}Coord;

// функция перемещает точку относительно текущей на deltaX по координате x
// и на deltaY по координате y
// изменения касаются переменной point - указатель на имеющуюся точку.
void move(float deltaX, float deltaY,Coord* point);
```

4. Круг задается координатами центра и радиусом. Описать структуру **Circle**. Реализовать функции: установки значений круга (координаты, радиус), подсчет площади круга.

```
typedef struct{
  int x, y;
  float r;
}Circle;

// Устанавливает значения. При этом происходит проверка допустимости значений:
//   допустимость значений с точки зрения существования токого круга.
// Если значения не допустимы, прерывается вся программа с сообщением "NO"
```

```
Circle setCircle(int _x, int _y, float _r);

// В качестве параметра передается переменная типа Circle. Возвращается площадь.
// Считаем, что значения полей заданы
// функцией setCircle  и, значит, такой круг существует.
float getSquare(Circle);
```

5. Треугольник задается длинами сторон. Описать структуру **Triangle**, которая содержит длины сторон в качестве полей. Реализовать функции задания размеров сторон треугольника, подсчета периметра и сравнения двух треугольников (геометрическое равенство треугольников).

```
typedef struct{
  int size1,size2,size3;
}Triangle;

// Устанавливает значения сторон треугольника.
// Если такой треугольник не может существовать, функция
// возвращает 0, если может - 1
int setTri(int, int, int, Triangle*);

// Возвращает периметр треугольника (существование проверять не нужно)
int getPerimetr(Triangle);

// сравнение двух треугольников, если равны, то возвращается 1, если нет - 0
int cmp(Triangle, Triangle);
```

Что изменилось.

1. Все атрибуты объекта можно "хранить в одном месте" даже если они разных типов.
2. Можно написать и использовать функции, которые принимают и возвращают в качестве параметров структурные переменные (сразу несколько простых типов)
3. При программировании можно "мыслить" целими объектами: треугольник, время и др.

-- TatyanaOvsyannikova2011 - 09 Jun 2014