Технологии программирования

Часть 1

Направление:
Информатика и вычислительная техника
Инфокоммуникационные технологии и системы связи

Ст. преподаватель
кафедры ПИВТ
Петрова О.Б.

2017 год

Разделы курса

- Основы объектно-ориентированного программирования и язык C++
- Библиотеки языка C++.
- Основы конструирования программных систем.
- Программирование приложения с использованием базы данных.
- Системы коллективной разработки программного обеспечения.

Основная литература

- Современные методы программирования на языках С и С++ / Л.Б. Бузюков, О.Б. Петрова. - СПб.: Линк, 2008.
- Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения. Учебник для вузов. 4-е издание. Стандарт третьего поколения. СПб, Питер, 2012.
- Технологии программирования : лабораторный практикум / О. Б. Петрова . - СПб. : СПбГУТ, 2015.

Дополнительная литература

Павловская Т. А. - C/C++. Программирование на языке высокого уровня, СПБ: Питер, 2010. - 260 с.

Шлее М. Qt 4.5 Профессиональное программирование СПБ.: БХВ-Петербург, 2010. - 896 с.

Буч Г. - Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.:БИНОМ, 1998. – 558 с.

Фридман А.Л. – Основы объектно-ориентированной разработки программных систем. М.: финансы и статистика, 2000. – 190 с.

Инструментальные средства для лабораторных работ

1. Netbeans IDE 7.x (Windows, Linux, Mac OS X) и выше

   www.netbeans.org

2. Code::Blocks 12 и выше (Windows, Linux, Mac OS X)

   www.codeblocks.org

3. Microsoft Visual Studio (Windows)
   - Visual Studio .NET 2013
   - Visual Studio .NET 2012
   - Visual Studio .NET 2010
   - Visual Studio .NET 2008

Компиляторы

- gcc (Linux, Mac OS X) – для Netbeans и Code::Blocks ниже 12 версии, устанавливается отдельно
- MinGW (Windows) - для Netbeans и Code::Blocks, устанавливается отдельно+пакет MSYS (дистрибутивы Code::Blocks версии 12 и выше содержат все необходимые компоненты)
- Cigwin (Windows)

Дополнительное программное обеспечение

- Инструментальная среда Qt Creator и библиотека Qt .

  https://www.qt.io

- Subversion (svn) — система контроля версии программного обеспечения

  http://subversion.apache.org/

  http://tortoisesvn.net/ (TortoiseSVN)

Классификация языков программирования

- Машинно-зависимые (машинные коды, ассемблеры)
- Машинно-независимые (языки высокого уровня, ЯВУ)
  - **Императивные (процедурные)**: Fortran, Cobol, Си, Pascal, Basic
  - **Функциональные**: Lisp, Haskell, Erlang
  - **Логические (декларативные)**: Prolog, SQL
  - **Объектно-ориентированные**: Smalltalk, C++, Objective-C, Java, Object Pascal, Ruby

Язык программирования Си

- Создан в начале 1970х годов
- Стандарты ISO: 1990, 1999, 2011.
- Процедурный язык общего назначения, используется для системного программирования.
- Достоинства: переносимость, гибкость, мощность, лаконичность.
- Типы данных: статическая типизация, встроенные типы данных, создание типов данных пользователем (структуры).
- Недостатки: отсутствует автоматическое управление памятью.

Структуры в Си

1. Объявление типа структуры
2. Объявление структурной переменной

Объявление типа (тег структуры Person):

```
struct Person
{
    char name[30]; /*поле структуры*/
    int year; /*поле структуры*/
} ;
```

Объявление переменной (nick):

```
struct Person nick;
```

Объявление синонима типа для структуры

```
typedef struct Person
{
    char name[30];
    int year;
} PERSON;

PERSON ann;
```

Анонимный тип структуры

```
struct
{
    char name[30];
    int year;
} tom;
```

tom — имя перемной структурного типа

Выделено памяти: 30 + 4 = 34 (байта)

Инициализация структурной переменной

```
struct Person
{
    char name[30];
    int year;
} ;
struct Person stud1 = {"John", 1994};
struct Person stud2 = {"Ann", 1993};
```

Обращение к полям структуры

Обращение к полю структуры через имя переменной:

stud1.year stud2.name

Обращение к полю через указатель:

```
struct Person* pStud = &stud1;
printf("%s  %d", pStud->name, pStud->year);
```

Действия над структурами

Использование в выражениях полей структур:

```
stud1.year = 1992;
int age = 2011 - stud1.year;
strcpy(stud1.name, "Bill");
printf("Имя:%s  возраст:%d", stud1.name, age);
```

Передача в функцию структурной переменной

```
struct Person
{
    char name[30];
    int year;
} ;
void input_struct(struct Person* p);
void output_struct(struct Person p);
int main(void)
{
    struct Person ann;
    input_struct(&ann);
    output_struct(ann);
    return 0;
}
```

```
void input_struct(struct
Person* p)
{
    scanf("%s", p->name);
    scanf("%d",&p->year);
}

void output_struct(struct
Person p)
{
    printf("%s %d\n",
      p.name, p.year);
}
```

Массив структур

```
struct Person
{
    char name[30];
    int year;
} ;
struct Person mas[4];
```

Обращение к полю i-го элемента массива:

```
mas[i].year = 1996;
(mas + i)->year = 1993;
```

Передача в функцию массива структур

```
struct Person
{
    char name[30];
    int year;
} ;
void input_struct(struct Person* p, int n);
void output_struct(struct Person* p, int n);
int main(void)
{
    struct Person su11[25];
    input_struct(su11, 25);
    output_struct(su11, 25);
    return 0;
}
```

```
void input_struct(struct Person* p, int n)
{
    int i;
    for(i=0; i<n; i++)
    {
        scanf("%s", p[i].name);
        scanf("%d",&p[i].year);
    }
}

void output_struct(struct Person* p, int n)
{
    int i;
    for(i=0; i<n; i++)
    {
        printf("%s %d\n", (p+i)->name, (p+i)->year);
    }
}
```

Управление памятью программы

Области памяти программы:

- Сегмент кода
- Статическая память (сегмент данных)
- Стек
- Динамическая память (куча)

Динамическое распределение памяти в C

- malloc

```
void* malloc(size_t size);
```

- calloc

```
void* calloc(size_t num, size_t size);
```

- realloc

```
void* realloc(void* ptr, size_t size);
```

- free

```
void free(void* ptr);
```

Пример создания динамической переменной

```
int *iPtr ;
iPtr = malloc(4);      /* malloc(sizeof(int)) */
scanf("%d", iPtr);
printf("Number: %d\n", *iPtr);
free(iPtr);
iPtr = NULL;
```

Пример создания строки в динамической памяти

```
char buffer[200]; *stPtr;
scanf("%s", buffer);

int len = strlen(buffer);

stPtr = malloc(len+1);

strcpy(stPtr, buffer);

printf("String: %s\n",stPtr);

free(stPtr);

stPtr = NULL;
```

Проверка выделения памяти

```
double* ptr;
ptr = (double*) malloc(max * sizeof (double));
if (ptr == NULL) printf(" Ошибка выделения памяти!");
else
{
        /* вычисления */
        free (ptr);
}
```

Еще вариант контроля

```
#include <stdlib.h>
int main (void)
{
    double* ptr;
    int max = 4;
    if ((ptr = (double*) malloc(max * sizeof (double))) == NULL)
    {
            printf(" Ошибка выделения памяти!");
            exit (EXIT_FAILURE); /* stdlib.h EXIT_SUCCESS*/
    }
    // вычисления
    free (ptr);
}
```

Язык программирования C++

- Создан в начале 1980х годов.
- Стандарты 1998, 2003, 2011.
- Объектно-ориентированный язык общего назначения.
- Имеет код, частично совместимый с Си.
- Состоит из ядра и стандартной библиотеки (пространство имен std).
- Большое количество сторонних библиотек, расширяющих возможности языка (диалекты C++).
- Имена файлов с исходным кодом: *.cpp, *.cc.

Особенности C++, отсутствующие в Си

- Новые стандартные типы данных (bool, string).
- Шаблоны (templates).
- Операторы управления динамической памятью (new, delete).
- Ссылки.
- Пространства имен (namespace).
- Перегрузка функций, операторов.
- Обработка исключительных ситуаций.
- Стандартные классы и объекты для организации ввода/вывода (cin, cout).

Пространства имен

- ключевое слово
- оператор разрешения области видимости
- глобальный идентификатор:
- идентификатор из пространства имен стандартной библиотеки C++:
- идентификатор из пространства имен пользователя:
- предложение using:

```
namespace
::

myfn();
std::cout << "1234";


myspace::myfn();

using namespace std;
cout << "1234";
```

Создание пространства имен

```
#include <iostream>
namespace my1{
    int a;
    int mult(int n);
}
namespace my2{
    int a;
    int mult(int n);
}
int my1::mult(int n){
    return n*2;
}
int my2::mult(int n){
    return n*3;
}
int main(){
    my1::a=10;    my2::a=12;
    std::cout <<"namespace my1: " << my1::mult(my1::a) <<std::endl;
    std::cout << "namespace my2: " << my2::mult(my2::a)<<std::endl;
    return 0;
}
```

Параметры функции

1. Передаются через стек
2. Виды параметров:
    - Параметр-значение,
    - Параметр-указатель,
    - Параметр-ссылка (в C++).

Передача параметра-значения

```
float mult(float a)
{
    a = 2*a;
    return a;
}
int main()
{
    float num = 15.5;
    printf("%.3f \n", mult(num));
    printf("%.3f ", num);
    return 0;
}
```

Ответ:

31.000
15.500

Передача параметра-указателя

```
void mult2(float* pa)
{
    *pa = *pa * 2;
}
int main()
{
    float num = 15.5;
    mult2(&num);
    printf("%f\n",num);
    return 0;
}
```

Ответ:
31.000

Передача параметра-ссылки

```
void mult3(float &b)
{
    b = b*2;
}
int main()
{
    float num = 15.5;
    mult3(num);
    printf("%f\n",num);
    return 0;
}
```

Ответ:
31.000

Создание динамических переменных в C++

Операторы C++:

1. new — выделение динамической памяти для одной переменной
2. new[ ] — выделение динамической памяти для массива
3. delete — освобождение динамической памяти из-под переменной (кроме массива)
4. delete[ ] - освобождение динамической памяти из-под массива

Пример программы с динамической переменной

```
#include <iostream>
using namespace std;
int main()
{
    float *iptr;  // объявление переменной-указателя
    iptr = new float; // выделение динамической памяти
    cin >> *iptr; // ввод числа в динамич. переменную
    cout << *iptr; // вывод из динамич. переменной
    delete iptr; // освобождение памяти
    iptr = NULL;  // iptr = nullptr для C++ 11
    return 0;
}
```

Массив в динамической памяти

```
#include <iostream>
using namespace std;
int main()
{
    float *mptr;
    int n = 3, i;
// выделение памяти
    mptr = new float[n];
// заполнение массива
for(i=0;i<n;i++)
{
        cout << "enter->";
        cin >> *(mptr+i);
}
// вывод массива
    for(i=0;i<n;i++)
        cout << *(mptr+i)<<endl ;
// освобождение памяти
    delete[ ] mptr;
    mptr = NULL;
    return 0;
}
```

Ввод/вывод в C++

- Консольный ввод/вывод — стандартные объекты-потоки cin (ввод) и cout (вывод):

```
std::cin >> a;
std::cout << "Текст " << a <<std::endl;
```

- Файловый ввод/вывод — классы ifstream и ofstream (подключить заголовок fstream), последовательность действий:
  - создать объект-поток,
  - открыть его в заданном режиме,
  - выполнить ввод/вывод данных,
  - закрыть объект-поток.

Форматирование потока std::cout

```
#include <iostream>
#include <iomanip>
#define N 3

using namespace std;

struct Tabl
{
    int number;
    string name;
    float hight;
};
```

Форматирование потока std::cout (2)

```
int main()
{
   struct Tabl mas[N]={

   1,"Peter",123.2785,2,"Tom",122.356,11,"Mike",134.15
   6};
    int i;
    cout<<"Вывод в поле определенной
   ширины"<<endl;
    for(i=0;i<N; i++)
    {

   cout<<setw(5)<<mas[i].number<<setw(6)<<mas[i].na
   me;
```

Форматирование потока std::cout (3)

```
cout<<"Вывод вещественного числа в формате";
cout<<" с плавающей точкой (научный)"<<endl;
cout.setf(ios_base::scientific,ios_base::floatfield);
for(i=0;i<N; i++)
{
    cout<<setw(5)<<mas[i].number<<setw(6)<<mas[i].name;
    cout<<setw(15)<<mas[i].hight<<endl;
}
cout<<endl;
```

Форматирование потока std::cout (4)

```
cout<<"Восстановление формата вывода";
cout<<" вещественного числа с фиксированной точкой";
cout<<endl;
cout.setf(ios_base::fixed,ios_base::floatfield);
cout<<"и определение точности вывода"
cout<<" вещественного числа"<<endl;
for(i=0;i<N; i++)
{
        cout<<setw(5)<<mas[i].number<<setw(6)<<mas[i].name;
        cout<<setw(10)<<setprecision(i+2)<<mas[i].hight<<endl;
}
cout<<endl;
```

Форматирование потока std::cout (5)

```
cout<<"Задание восьмеричной системы счисления";
cout<<" для вывода целого числа"<<endl;
for(i=0;i<N; i++)
{
      cout.setf(ios_base::oct,ios_base::basefield);
   //oct,dec,hex
      cout<<setw(5)<<mas[i].number;

      cout.setf(ios_base::dec,ios_base::basefield);
      cout<<setw(6)<<mas[i].name;
      cout<<setw(10)<<setprecision(i+2) <<mas[i].hight<<endl;
}
cout<<endl;
```

Форматирование потока std::cout (6)

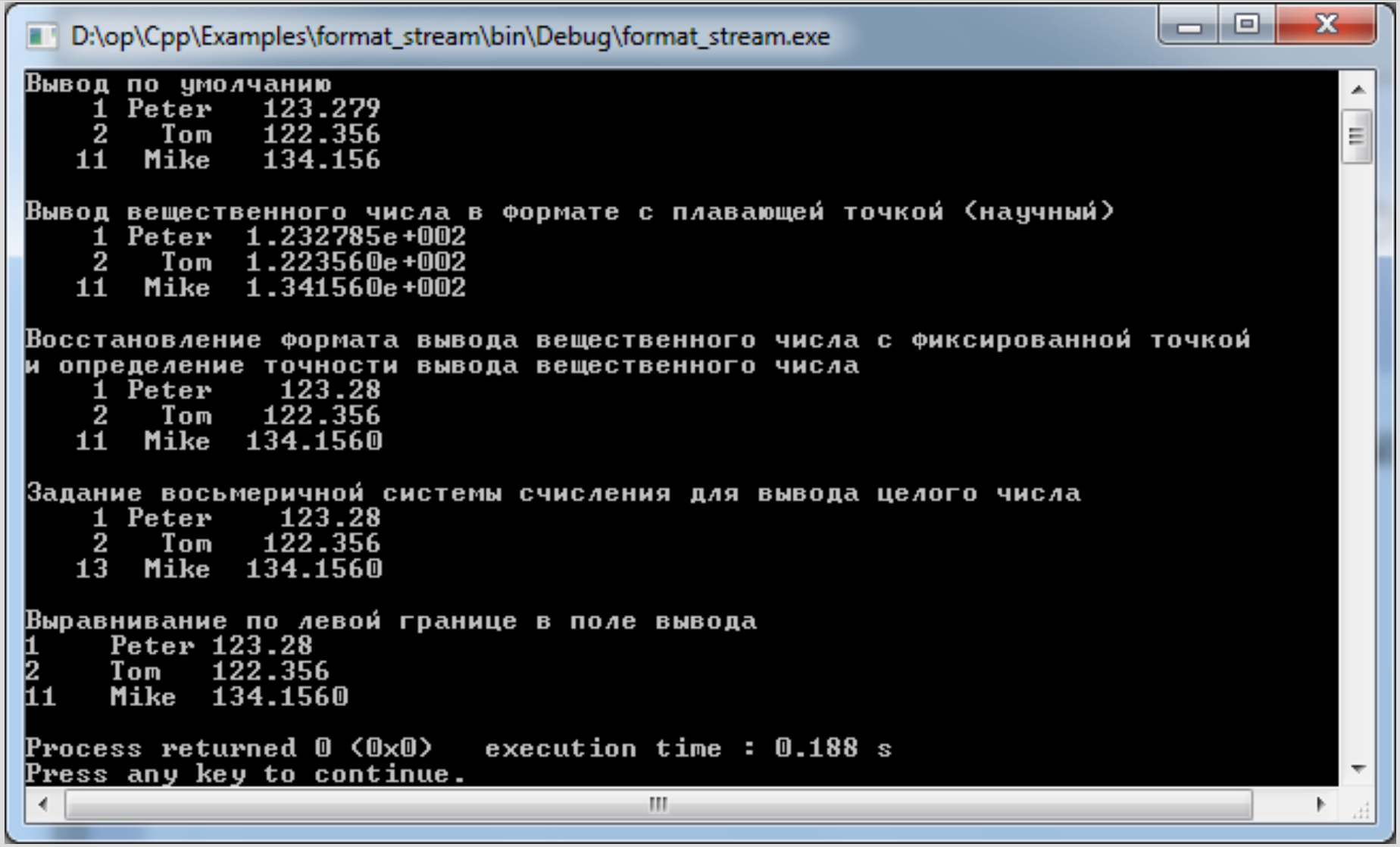
```
cout<<"Выравнивание по левой границе";
 cout<<" в поле вывода"<<endl;

 cout.setf(ios_base::left);
 for(i=0;i<N; i++)
 {
     cout<<setw(5)<<mas[i].number;
     cout<<setw(6)<<mas[i].name;

cout<<setw(10)<<setprecision(i+2)<<mas[i].hight<<endl;
 }

 return 0;
}
```

Форматирование потока std::cout (7)



Пример работы с файлом: чтение из файла

```
// Чтение из файла одного значения
std::ifstream fin;
fin.open("my1.txt");
if (fin)
{
	fin>>number;
	fin.close();
}
else
{
	cout << "file not found";
	exit(1); // подключить stdlib.h или cstdlib
```

Пример работы с файлом: запись в файл

```
// Запись в файл одного значения

std::ofstream fout;

fout.open("my2.txt");

fout <<"Number= "<< number<<"\n";

fout.close();
```

Класс string

```
#include <iostream>


int main(int argc, char** argv)
{
    std::string st1("My "), st2="string", st3;
    st3 = st1+st2;
    int i;
    for(i=0;i<st3.size(); i++) // st3.length()
            std::cout << st3[i] <<std::endl;
    return 0;
}
```

Другие методы класса string

insert — вставка подстроки

```
string st("ererer");
st.insert(1,"567"); //e567rerer
```

erase — удаление символов

```
st.erase(5,3); //e567rr
```

empty — проверка наличия символов в строке

```
st.clear();
if(st.empty())
   cout << "Пустая строка "<<ndl;
else cout << st<<endl;
```

c_str — получение указателя на строку

```
int len = strlen(st.c_str());
```