Информатика. Спецглавы

Лекция 1

Направление: Инфокоммуникационные технологии и системы связи
2012 год

Разделы курса

- Основы объектно-ориентированного программирования и язык C++
- Основы Web-технологий
- Основы конструирования баз данных

- 10 лекций
- 17 лабораторных работ

Литература

- Современные методы программирования на языках С и С++ / Л.Б. Бузюков, О.Б. Петрова. - СПб.: Линк, 2008
- WEB-технологии. / Бузюков Л.Б. - СПб.: СПбГУТ, 2011.
- Дополнительно:
- Буч Г. - Объектно-ориентированный анализ и проектирование с примерами приложений на С++. М.: БИНОМ, 1998. – 558 с.
- Фридман А.Л. – Основы объектно-ориентированной разработки программных систем. М.: финансы и статистика, 2000. – 190 с.

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

Проверка выделения памяти

```
double* ptr;
int max = 4;
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);
}
```

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;
```

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++).

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 uses:

namespace

::

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

myspace::myfn();

uses namespace std;
cout << "1234";

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
2. delete — освобождение динамической памяти из-под переменной (кроме массива)
3. 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;
    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::cout << " " << std::endl;
std::cin >> a;
```

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

Пример работы с файлом

```
// Чтение из файла
std::ifstream fin;
fin.open("my1.txt");
if (fin)
        fin>>number;
fin.close();
// Запись в файл
std::ofstream fout;
fout.open("my2.txt");
fout <<"Number= "<< number<<"\n";
fout.close();
```

Тип 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;
}
```