Atividades aula 2

Fluxo de projeto

Prof. Rafael Corsi Ferrão
10 de março de 2016

Rafael Peres Gobo – R.A.: 11.03055-0
Victor Henrique Salgueiro Dias – R.A.: 12.01022-7
Kaio Alexandre Strippoli Doffini – R.A.: 12.01661-6

1. Fluxo de Projeto

Realize uma pesquisa sobre os seguintes tópicos:

a. O que é:

- **Compilador C**
- **Assembler**
- **Linker**

Compilador C *é um programa que traduz um código de programa escrito pelo usuário em linguagem C (escrita essa que é de fácil entendimento/compreensão por parte deste – linguagem de "alto nível) para a linguagem simbólica de máquina (assembly) para um processador, que é semanticamente equivalente à linguagem de alto nível.*

Assembler *é o programa que realiza a transformação da linguagem simbólica assembly, efetivamente, para a linguagem de máquina, bem como a transformação de variáveis em códigos binários e suas alocações em seus respectivos espaços de memória.*

Linker *é uma espécie de ponte entre elementos gerados pelo compilador - como parâmetros e bibliotecas utilizadas no programa, por exemplo – com o arquivo executável final.*

b. O que é um RTOS? Descreva uma utilização.

RTOS *(Real Time Operating System – Sistema Operacional de Tempo Real) é um sistema operacional comumente utilizado para aplicações de medição e controle, uma vez que sua característica é gerenciar os recursos de hardware e aplicativos que são rodados num computador com altíssima precisão e grau de confiabilidade. Um exemplo é a utilização de um RTOS em monitores cardíacos: a sua resposta deve ser dada de forma rápida (por meio de sinais sonoros, por exemplo), após constatar que houve variação brusca nos batimentos cardíacos de um paciente.*

c. O que é desenvolvimento em V (Modelo V)?

Modelo V *é um conceito de Engenharia de Sistemas de resolução/melhoria de problemas em desenvolvimento de sistemas que possui como característica o fato de que os testes sejam feitos contra os próprios requisitos do componente ou interface no qual está sendo realizado o teste. Assim, esse modelo conceitual permite a refinar os erros encontrados durante o processo de derivação da especificação dos requisitos, além de possibilitar o desenvolvimento de novos requisitos do sistema.*

d. Qual a diferença entre C e C++?

As duas linguagens possuem algumas características semelhantes: ambas são compiladas, de alto nível e portáveis. Porém, é possível listar diversas diferenças entre elas: aplicações em ***linguagem C*** *são mais rápidas para compilar, bem como para executar, em relação à aplicações em* ***linguagem C++****. Esta, por sua vez, gera um resultado bastante eficiente, já que a linguagem C está contida na linguagem C++, o que torna a última mais completa – há mais recursos e funções. Além do mais, a linguagem C++ é multi-paradigma, o que permite uma programação com paradigma Estruturado ou Orientação a Objetos. Já a linguagem C possibilita a programação apenas em paradigma Estruturado. Em suma, por ser uma linguagem mais completa e por ser uma extensão da outra, a linguagem C++ é recomendada para aplicações relativamente grandes, com difícil controle e manutenção do código.*

2. Revisão C

2.1 Printf_A

Crie um programa que exiba a seguinte saída :

Eletronica Embarcada
Aula 2 - Revisao C
#seu nome, #data

Arquivo: 2.1 printf_A.c

2.2 Printf_B

Crie um programa que exiba um contador de 0 até 100 separado por "; ":

0; 1; 2; 3; ...

Arquivo: 2.2 printf_B.c

2.3 Printf_C

Crie um programa que exiba os vinte primeiros números primos listados a seguir :

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71
Para isso, salve os valores em uma constante e faça a varredura. Exibindo o seguinte resultado :

Primo 1 = 2
Primo 2 = 3
Primo 3 = 5
Primo 4 = 7

Arquivo: 2.3 printf_C.c

2.4 Função 1

O seguinte programa calcula o maior divisor comum entre dois números, extraia a parte que calcula o MDC e crie uma função para ela. A parte de exibição dos dados não deve estar dentro da função.
Reescreva o programa utilizando a função criada. Ela deve retornar o MDC e receber os dois números.

```
/* C Program to Find Highest Common Factor. */
#include <stdio.h>
int main () {
int num1 , num2 , i , hcf;
printf("Numero 1 :\n");
```

```
scanf( "%d" , &num1);
printf("Numero 2 :\n");
 scanf("%d" , &num2);
for(i =1; i<=num1 | | i<=num2 ; ++i ) {
 /* C h e cki n g w h e t h e r i i s a f a c t o r o f b o th number */
if ( num1%i==0 && num2%i ==0)
hcf=i ;
}
printf( "H.C. F o f %d and %d i s %d \n" , num1 , num2 , hcf) ;
return 0 ;
}
```

Arquivo: 2.4 funcao_1.c

2.5 Função 2

O que é a prototipagem de funções em C? Qual a sua utilização?

Prototipagem de função em um código de programa em linguagem C é, basicamente, a declaração de funções antes da função *main* do programa. Em outras palavras, fazer o protótipo de uma função significa declarar que esta função existe no programa e seu código está em alguma linha específica. Uma vez que a função tem seu protótipo no programa, ela não precisa ter seu código escrito antes da função *main*, o que deixa o código do programa, em si, mais limpo e de fácil entendimento, já que o uso do protótipo de funções possibilita que a função *main* seja a primeira função do código.

2.6 Ponteiro 1

Qual deve ser a saída do trecho de código a seguir :

```
int count = 10 , *temp , sum = 0 ;
temp = &c o u n t ;
*temp = 2 0;
temp = &sum ;
*temp = count;
printf("count = %d , *temp = %d , sum = %d\n" , count , *temp ,sum);
```

A saída do código deve ser:
*count = 20,*temp = 20, sum = 20*

Abaixo é explicado através de comentários, como chegamos no valor de saída:

```
int count=10,*temp,sum=0; //declaração de variáveis

temp = &count; //ponteiro temp aponta para o endereço de memória da variável counte; *temp = count = 10 ; sum = 0
```

**temp = 20; //o valor contido no endereço de memória sendo apontado por temp recebe o valor 20; *temp = count = 20 ; sum = 0*

*temp = ∑ //ponteiro temp aponta para o endereço de memória da variável counte; count = 10 ; *temp = sum =0*

**temp = count; //o valor contido no endereço de memória sendo apontado por temp recebe o valor count = 20; *temp = sum = count = 20 ;*

*printf("count = %d,*temp = %d,sum = %d\n",count,*temp,sum); //count = 20, *temp = 20, sum = 20;*

2.7 Ponteiro 2

Escreva uma função swap(), com a seguinte configuração :
void swap(int ∗p1, int ∗p2);

Use essa função para trocar o valor entre duas variáveis.

```
int main() {
int x = 10;
int y = 20;
// Voce deve c o n f i g u r a r e s s a f u n ca o c o r r e t am e n t e swap ( . . .
)
//Deve imprimir na tela x: 2 0 , y: 10
printf("x: %d , y: %d\n" , x , y ) ;
}
```

Arquivo: 2.7 ponteiro_2.c

2.8 Ponteiro 3

De acordo com o padrão C, o acesso a índices de vetores feitas por : arr[0] está errado, a forma correta é *(arr+0).
Analise a afirmação anterior e escreva um programa que acessa um vetor pela notação correta e imprima os seus valores.

Arquivo: 2.8 ponteiro_3.c

2.9 Ponteiro 4

Crie uma função : print_array que irá imprimir na tela todos os valores de uma array de inteiros (int test_array[20]). Quais são os parâmetros corretos de entrada dessa função que deve ser utilziado para o correto funcionamento ?

Os parâmetros de entrada dessa função devem ser: o endereço de memoria do array e o comprimento desse array, como mostrado abaixo:

print_array(&test_array, sizeof(test_array)/sizeof(test_array[0]));

Arquivo: 2.9 ponteiro_4.c

2.10 Ponteiro 5

A seguinte função promete devolver um vetor com os 3 primeiros números primos maiores que 1000 e devolver o endereço da nova lista.
Onde está o erro?

```
int *primos( void )
{
int v[ 3 ];
v[ 0 ] = 1009; v[ 1 ] = 1013; v[ 2 ] = 1019;
return v;
}
```

O erro está no retorno da função. Como a função é do tipo ponteiro, ela deve consequentemente retornar um ponteiro, porém o retorno da mesma é uma variável local da função a qual é "liberada" pelo sistema após a execução da função. Ou seja, após o retorno da função a variável "v" não existe no programa e a memória utilizada por ela durante a execução da função foi liberada, não sendo possível acessá-la.
Para isso precisa-se utilizar o comando malloc() para alocar os dados em questão em um endereço de memória permitindo que a função principal consiga acesá-los.

Arquivo: 2.10 ponteiro_5.c

2.10 Malloc

O que a função malloc() faz ? Qual outra função deve ser utilizada após sua utilização ?

A função malloc() aloca os dados da variável em um endereço físico de memória permitindo o acesso à esses dados de forma global, mesmo tratando de valores referentes a variáveis locais.
Após ser utilizada a função malloc(), é necessário utilizar a função "free()" para liberar a memória utilizada.